Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/12362
DOI URL: http://dx.doi.org/10.21474/IJAR01/12362

RESEARCH ARTICLE

ENHANCED ALGORITHM OF ARTIFICIAL BEE COLONY (ABC) TO OPTIMIZE MODELS OF SYSTEM RELIABILITY

Samuel Acquah[1], Li Zhen[2] and Anastasia Krampah-Nkoom[3]

1. Mechatronics & Automation Engineering Department, Shanghai University, China.
2. Electronics & Information Engineering Department, Jiangsu University of Science and Technology, China.
3. School of Management and Business Administration, Jiangsu University, China.

Manuscript Info

Manuscript History
Received: 25 November 2020
Final Accepted: 28 December 2020
Published: January 2021

Key words:-
Software Reliability, Software Quality, Metrics, Reliability, ABC, C4.5

Abstract

In recent times, computer software applications are increasingly becoming an essential basis in several multipurpose domains including medicine, engineering, transportation etc. Consequently, with such wide implementation of software, the imperative need of ensuring certain software quality physiognomies such as efficiency, reliability and stability has ascended. To measure such software quality features, we have to wait until the software is executed, tested and put to use for a certain period of time. Numerous software metrics are presented in this study to circumvent this long and expensive process, and they proved to be awesome method of estimating software reliability models. For this purpose, software reliability prediction models are built. These are used to establish a relationship between internal sub-characteristics such asinheritance, coupling, size, etc. and external software quality attributes such as maintainability, stability, etc. Usingsuchrelationships, one canbuildamodelinordertoestimatethereliabilityofnewsoftware system.Suchmodelsaremainlyconstructedbyeitherstatisticaltechniquessu chasregression,or machine learningtechniquessuchasC4.5andneuralnetworks.The prototype presented isinvigoratedemployingprocedures of machine learninginparticularrule-basedmodels.Thesehaveawhite-boxnaturewhich accordsthecataloguingandmakingthemgood-looktoexpertsinthedomain. In this paper, wesuggest a powerfulinnovative heuristic based on Artificial Bee Colony (ABC) to enhance rule-based software reliability prediction models. The presented approach is authenticated on data describing reliability of classes in an Object-Oriented system. We compare our models to others constructed using other well-established techniques such as C4.5, Genetic Algorithms (GA), Simulated Annealing (SA), Tabu Search (TS), multi-layer perceptron with back-propagation,multi-lay perceptron hybridized with ABC and the majority classifier. Results show that, in most cases, the propose technique out- performs the others in different aspects.

Corresponding Author:- Samuel Acquah
Address:- Mechatronics & Automation Engineering Department, Shanghai University, China.

Introduction:-

It is known that, evaluating the quality, functionality and reliability of a software system has been a major concern in software quality setting. To predict software reliability is quite difficult.The main problem is apprehensive primarily with design faults, which is a very dissimilar situation from that handled by conventional hardware theory. A fault can be defined as manifestation in the code of an error made by the designer with respect to the requirements of the software system. Activation of a fault of an input value leads to an incorrect output [1]. However, detection of such an event corresponds to an incidence of a software system failure. The input values to the software modules either internally or externally may be considered as arriving to the software randomly. In view of this, the software failure may not be generated stochastically, it may be detected in such a manner. Thus, it justifies the use of stochastic models of the underlying random process that administers the software failures [2]. Reliability of a software system can be explained as the probability of failure- free operation of a computer program for specific period of time in a specified environment. Statistical calculations and parameter estimations of software reliability is crucial tool for developing reliable software systems.

Moreover, other researchers proposed numerous computer programs which are attested to be correct by this method but the program contained faults. Program testing is more practical approach and is empirical in nature. Program testing basically involves symbolic or physical execution of a set of test cases with the objective of exposing embedded faults in the program [3]. However, copious reliability software algorithms and metrics have been engineered and implemented by computer programmers to estimate the parametersofsoftwarereliabilitymodels.AnimprovedArtificialBeeColony(ABC)algorithm was projected to estimate the parameters of software reliability model [4]. This Algorithm has the capacity of dualistic search which makes the algorithm has more powerful worldwideexploration and better performance. Similarly, Particle Swarm Optimization (PSO) Algorithm to optimizethe problem of software reliability growth model and predict the number of software failures [5]. The performanceandaccuracyofABCalgorithmwillbeexaminedonthenumericproblemswithmulti-dimensionalandcomparewiththePSOalgorithm.However, thisresearchfocusesonhybridABC, GAandPSOalgorithmsandhowtheyareusedtooptimizevarioussoftwarereliabilitymodels.Go- model parameter is chosen as representative event function with respect to ABC and PSO hybrid algorithms.

Research Background:-

It is obvious that, computer engineering and technology largely play significant role in our contemporary times across the ages of human endeavors today. An extensive range of highly multifaceted software systems are progressively all-pervading in the areas of, Agriculture, Aerospace, Industrial control operations, Military operations, Processing industries, Transport operations, Finance, Health, and other related fields, playing an increasingly role. It is for this motive that, the quality performance of these complex software must be assured at all cost. However, the quality of a software product decides its acceptance or fate in the software development algorithm[6].

Through theoretical research and engineering practice, many scientist and computer engineers by their grace wisdom and effort, numerous software reliability models have been proposed to predict and assess the reliability of software system as well as detecting its functional status. These models help the designer to do quantitative analysis and predict the credibility and behavior of the software before the release of the software to the world market. Some of these models include: Goel-Okumoto Model, Yamada S-Shaped Model, the Go model, a Weibull model, MO model, JM model, White-box and Black –box model. These models ensure the quality and reliability of the software. However, most of these models are in non-linear function which implies that, it is very difficult to determine their optimal parameters[7]. Usually, Maximum Likelihood Estimation (MLE) and Least Square Estimation (LSE) methods are two traditional techniques commonly used to estimate parameters. This approach is experimented by observations and inspirations of special behavior and collective movement of colonies of honey bees, swarm of ants, shoal of fish, and flock of birds[8]. However, the most common intelligent optimization algorithms include: Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), ant Colony Optimization (ACO), artificial fish-swarm algorithm (AFSA). These methods arestochasticoptimizationtechniquesthathavebeenusedbyengineersinwiderangeofnumerical application functions to solve real world optimizationproblems.

Software reliabilityEstimation
Software reliability is a point to which a component, software system or process meets the neededrequirementswithoutsystemfailure.Itismeasuredbyitscharacteristics.

Functionality:
It is the ability of a software system to provide functions which meet stated and implied needs when the software is used under stated condition.

Reliability:
It is the capacity of a system or component to perform its required functions under stated conditions for a definite period of time.

Maintainability:
The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.

Portability:
"It is defined as the ease with which a system or component can be transferred from one hardware or software environment to another.

The reliability of a software compliance is defined as the ability of the software product to stick to standards, conventions or regulations relating to reliability. Hence,softwaredefectcanberegarded as an indicator of software reliability.[9].

Table 1:- Detailed characteristics and sub-characteristics of software reliability.

Characteristics	**Sub-characteristics**
Functionality	Suitability, portability,interoperability,security, functionality compliance
Reliability	Maturity, fault tolerance, recoverability, reliability compliance
Usability	Understandability learnability, operability, attractiveness, usability compliance
Efficiency	Time behavior, resource utilization, efficiency compliance
Maintainability	Analyzability, changeability, stability, testability, maintainability compliance
Portability	Adaptability, install ability, co-existence, replace ability, portability compliance

To evaluate software quality and its reliability, the software system must first be executed, comprehensivelyassessedbeforeputtouse.Thisunbearablylongsoftwarelifecyclecanbeavery perilousandcostly.Additionally,thetestingstageinitselfisthemostessential.Becauseerrorsare unavoidable in software development; around "40 to 50% of user programs contain nontrivial faults[10] . It is stated that, simply testing the software would require at least 50% of the cost of development. It might even cost more in the case of safety critical software. Also, Dueto softwarefailure,theUSDepartmentofDefenselosesmorethanfour billiondollarsperyear[11]. Duetosuchevidence, it ismostimportanttoassesssoftwarequalityanditsreliability. This is why copious software reliability metrics have been presented in this study, such as McCabe's cyclomatic metric. Halstead's software science metric [12].

Table 2:- Proposed metrics.

CBO	Coupling Between Object Classes
LCOM	Lack of Cohesion in Methods
RFC	Response for a class
WMC	Weighted Methods per class
DIT	Depth of the class in the Inheritance Tree

These metrics are utilized to assess internal software reliability physiognomies. To emphasize the essential of measuring, we made reference to Pressman's quote "if you don't measure, judgment can be based only on subjective

evaluation"[11]. With measurement, trends can be spotted, better estimation can be made, and true improvement can be accomplished over time [13]

Software Reliability Estimation Models

Due to the fact that, we unable to evaluate the reliability of software attributes, we depended on estimating them. As a result, we utilized estimation/prediction models such as mathematical/statistical. In this study, we principally focused on logical models since they can easily be interpreted. To predict the stability of software system, let us cogitate the vector as data point in the form V= V = {, …,t n,…c} where ti represents metric and c label of classification (0=stable, 1 not stable).However,theclassinanObject–Orientedsystemisdefined by a data point. The table below displays data set of 14 cases, four metrics; these include Number of Children ((NOC), Lines of Code (LOC), Lines of Comments (LOC) and Number of Public Methods (NPM) and one label of classification.

Table 3:- Software stability data set of classes in an Object-Oriented system.

DATA SET	NOC	LOC	LOC	NPM	FIRMNESS
1	5	801	810	2	1
2	5	490	1200	1	1
3	3	4600	4040	2	0
4	2	130	16000	2	0
5	2	701	490	2	0
6	2	202	120	5	1
7	3	49	201	0	0
8	5	124	1003	2	0
9	5	760	1202	2	1
10	2	340	490	6	0
11	4	469	2900	2	1
12	2	3481	3460	2	0
13	2	1602	50400	1	1
14	3	7501	40500	2	1

The connection between the unknown classification label and the metrics is proven by thesoftwarepredictionmodels.However,oneofsuchmodelsisdecisiontree. Thismodeliswidely used to predict the reliability of software models. What below is a decision tree deduced from the abovedataset.Usually, thedecisiontreeisinversebeginningwiththerootnodedowntoaleaf. If a leaf is reached, then the classification label can be defined. In the process, nodes encrypt tests andonentirepath isaconjunctionofsuchtests.Thetreereads, "IfNOCismorethan3andLOCOis more than 410, then the class is notstable.

The example of decision tree below can grow and become relatively difficult to read by human experts. It can also be converted into rule set. A rule set is a logical grouping of code analysis rules that identify targeted issues and specific conditions. From the above decision tree, a rule set can be deduced as follows. Rule 1: NOC>3&LOCO>410, Rule 2: NOC≤2&LOC≤200, Default class: 0. The rule set can be interpreted as, (NOC) denotes the number of children which is greater than 3 and lines of comments (LOCO) also greater than 4100, then it unstable class. However, if the class has NOC less than or equal to 2 and lines of code (LOC) smaller than or equal to 200 then it not stable. Indeed, this rule is mostly used because it has the ability to serve as a guide in building a class with particular software reliability attribute. The reliability ofsuch rule setsis evaluated by accuracy, error rate, the balanced accuracy, Sensitivity, Specificity, and Precision. Confusion matrix for binary classification is used to display these evaluation measurements.

Figure 2:- Confusion matrix for binary classification.

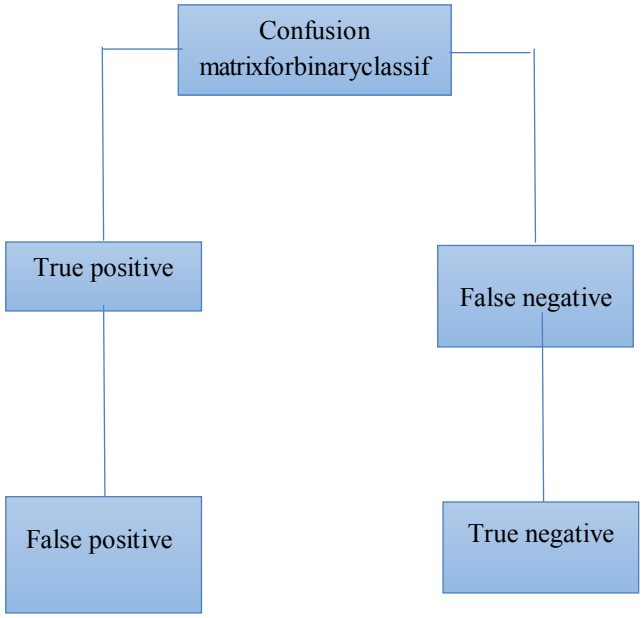
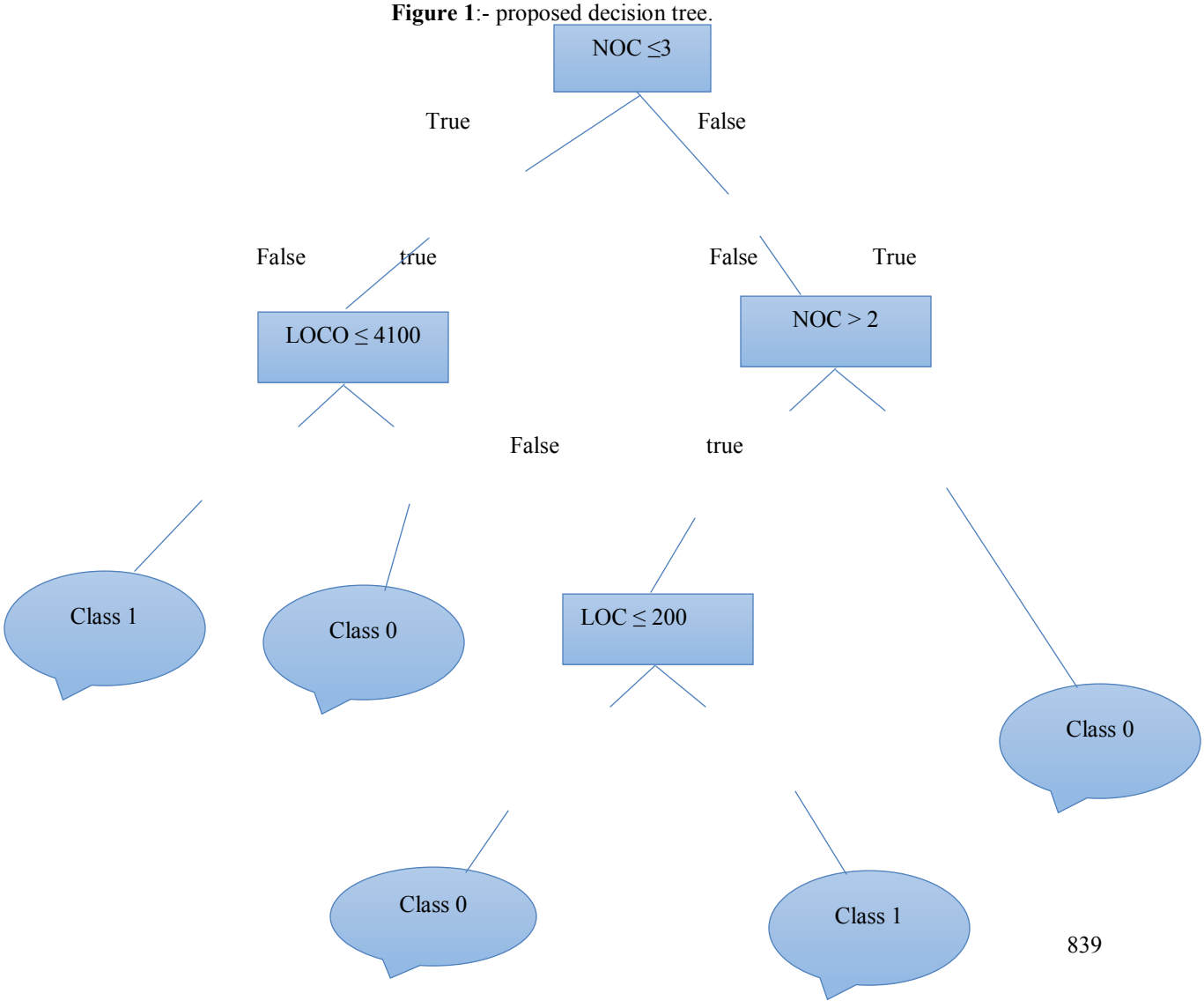
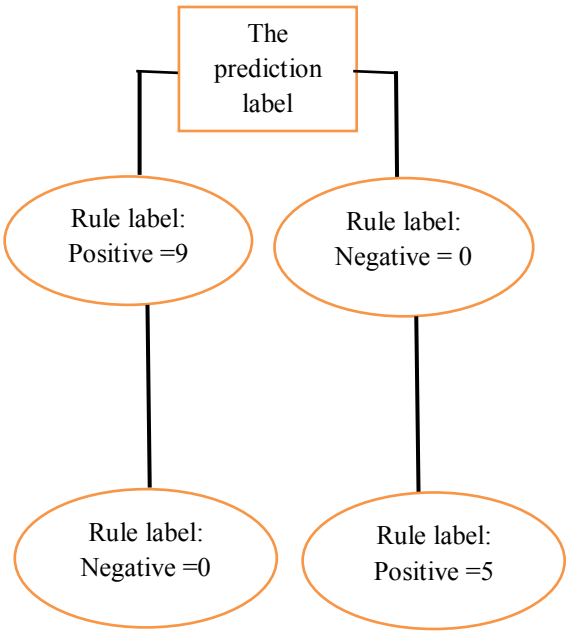


Figure 1:- proposed decision tree.



In every entry c[y][t], we recorded the number of classesthat were identified by the model with labely whiletheir real classification is**t**. We defined a positive classlabel as a stable class (label0) and a negative classlabel asunstable class (label1). **True Positive**: denotes the number of classesthatare positive and were classified assuch. **FalseNegative**: represents the number ofclasses that are positive butwere categorized as negative. **False Positive**: indicates the number of classes that are negative but were categorized as positive**True Negative**: signifies the number of classesthat are negative and were categorized as such. Gaging the rule set shown above, the researcher acquired the confusion matrix as displayed above. However, the Rule 1 applies to data set 1, 2, and 8. Rule 2 applies to data set 6 and 14, while as the default rule applies to data set 3, 4, 5,7, 9, 10, 11,12 and 13. Quite a number of measurement techniques have been employed to evaluate the data set cases in this regard. In the assessment of Rule set using confusion matrix, we obtained.

Figure 3:- Assessment of Rule set using confusion matrix.



Mathematically, we let N= be the estimation model. This expression denotes the accuracy of {N}. However, this expression computes the percentage of data set which are accurately identified by N.

$$\text{Accuracy}\{N\} = \frac{\text{truepositive} + \text{truenegative}}{\text{truepositive} + \text{falsnegative} + \text{true negative} + \text{falsepositive}} \quad (1)$$

$$A\{N\} = 1 - \text{Accuracy}\{N\} \quad (2)$$

Eq (2) denotes the error rate of {N}, whereas A {N}, implies that the data set is falsely classified by N.

From the above equations, it can be inferred that, there is inaccuracy due to data set imbalance. Below is an equation denoting a balanced accuracy of [n]. Here, we contemplated the case of a data set containing 100 data cases. Ninety-nine data set have a classification label 0, and 1 data case has a classification label of 1. In this vein, a model assigning label (0) to all the cases has an extremely high accuracy (close to 100%).

The situation arises when the misclassification of the fewer frequent categorization label is more expensive, that is when class label (1) in the previous example refers to an unhealthy individual but the model is classifying this case as a healthy individual who will not start his or her desired treatment. In this case, accuracy is not the appropriate measure to contemplate. Instead, the balanced accuracy computes the average accuracy of the classification model therefore giving equal weight to both classification labels.

Eq.3: displays thebalanced accuracy

$$\text{Balanced accuracy}\{N\} = \frac{1}{2} * \frac{\text{true positive}}{\text{true negative} + \text{false positive}}$$

$$= +\frac{1}{2} * \frac{\text{true negatve}}{\text{truenegative} + \text{falsepositive}}$$

Here, it can be noted vividly that, Eq.3 denotes sensitivity because of its balanced accuracy whereas the second section denotes specificity. These two measurement measures have been signified by Eq.4 and Eq.5

$$\text{Sensitivity } \{N\} = \frac{\text{truepositive}}{\text{truepostive} + \text{falsenegative}} \quad (4)$$

$$\text{Specificity } \{N\} = \frac{\text{truenegative}}{\text{truenegative} + \text{falsepositive}} \quad (5)$$

Another famous measurement measure is precision. This measurement technique has the ability to show the probability of true positive cases. Many scientists and engineers prefer this technique when dealing with cases of reliability predictions. The prediction is computationally expressed as.

$$\text{Precision } \{N\} = \frac{\text{truepositive}}{\text{truepositive} + \text{false}} \quad (6)$$

Figure 4:- To computationally display these measurements methods, we used confusion metrics as shown below.

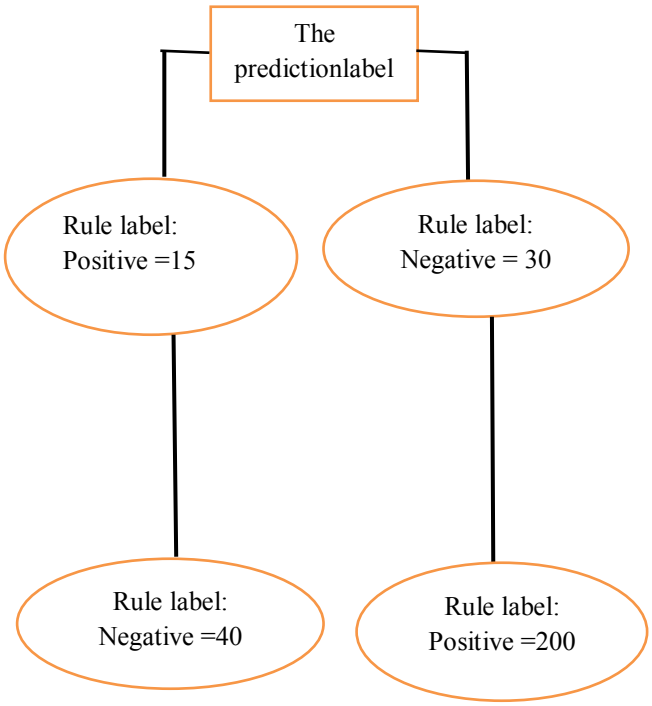


Table 4:- Software reliability measurement calculated from figure four.

Software measurement function	value
Balanced accuracy	0.5*(15/(15+30))+0.5*(200/200+40) ≈ 20.67
Error rate	1−20.67 ≈- 19.67
sensitivity	15/(15+30) ≈ 0.3333
Specificity	200/(200+40) ≈ 0.8333
precision	15/(15+40) ≈ 0.2727
correction	(15+200)/285 ≈ 0.7544

Statistical Models

In the domain of software reliability parameter estimation, statistical model has been largely rummage-sale worldwide by research scientists and engineers. These models are constructed by utilizing discriminant analysis, principle component, regression techniques etc. according to[14], a discriminant analysis method for identifyingdefective computer programs was proposed. The main goal is to decrease the preliminary set of metrics into a subclass of non-correlated metrics. [15]Came up with multiple linear regression. They employ convolution metrics as indicators of change prone software systems. [16]Emerged with two subsystems of multi-purpose operating system. [16]Proposed with two subsystems of multi-purpose operating system. [17]Proposed medical imaging system that is used as experimental statistics. [18] Proposed relative least square and minimum relative error. They equated their models to least square and least absolute value. Minimum relative error appeared to be more powerful than other techniques only when data is approximately normally distributed. The authors notice a substantial enhancement in predicting software changes during the maintenance.

[19]Investigated five major object-oriented software metrics recommended by Chidamber and Kemerer (1994). They also emerged with additional five objected oriented metrics. Their objective is to construct two least – square regression models whose dependent variable is the maintenance exertion well – defined asthe number of lines changed in a class. They examined with many variations of the model by considering a dissimilar subset of the independent variable every time.

Data is collected from two profitable software systems written in an object – oriented programming language. The two software systems include interface system and quality evaluation system. However, experimental results show the effectiveness of the metrics used in predicting the maintenance exertion and emphasize the essential of size (1) and size (2) as estimation metrics, where size (1) denotes the total number of semicolons in a class and signifies the absolutenumber of attributes and local procedure in a class.

HeuristicsandMetaheuristics

This portion briefly presents related works of heuristics and metaheuristics. These include; GeneticProgramming (GP), Genetic Algorithm(GA), Simulated Annealing(SA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO)and Artificial Bee Colony(ABC).

Genetic Programming (GP).

In the paradigm of software engineering, Genetic Programming is initially presented by [20]. The method of GP is to pose the improvement of software reliability modules. The approach is based on the explanations by [21]. Operator and cyclomatic complexity are two metrics used and are related lines of codes, (McCabe, 1976). These metrics are said to have robust relationship with reliability estimation proved by [22]. In this study, the assessment of the GP largely depends on the law of Pareto which denotes that "20% of the modules will classically account for approximately 80% of the errors" this method is said to be strong as opposed to random model when assessed on two major industrial assignments. These include; Pascal – written legacy telecommunication system and Ada – written command Control and Communications System (CCCS).[23]Emerged with GP method to mainly predict error – proneness and change- proneness using huge Windows – based applications coded in C++ programming language. To evaluate over – fitting, GP uses a random subclass selection of the data. GP is therefore evaluated on the whole data set given product and process metrics. The addressed GP is unique in that, it assimilates previous likelihood and misclassification into one ultimate fitness function. It is shown that, in comparison to logistic regression, Genetic Programming accomplishes awesome outcome with respect to Type -I and Type – II faults. Over improved 9% correctness is achieved by GP against logistic regression.

Genetic Algorithms

The use of Genetic Algorithm (GA) is introduced by [24] to specifically enhance the estimation of software strength. The strength of software is evaluated with respect to the size and domain specific values of the software. Experimental outcomes indicate that, GA has ability to enhance the precision of software estimation models. A comprehensive method to syndicate and acclimatize existing models to software systems from a particular domain is introduced by [25]. Their method was ably assessed on the Java classes of prediction stability. The results of the assessment indicate that, the introduction of two methods significantly perform better than C4.5. [26]Emerged with outstanding approach to optimize strength estimation models. In the approach, the author used Methodology

metrics and Developed Lines of Code (DLOC) together with the application of GA. The method is successfully validated in the application of 18 NASA data sets.

[27]Posed GA method to predict the stability of software system. The method syndicates only one model built on common domain systems to data from dissimilar domains. The method is found to be better than C4.5. Dhiman, Goyal and Sandhu (2009), came up with GA method and assess it on jEdit data set for error –proneness estimation. The authors used a metric from chidamber and Kemerer (1994) together with LOC and Number of Public Methods (NPM). The method proves to have 80.14% correctness.On GA - based models for software strength estimation. The author employs KLoC metric and yields in a valid estimation model when evaluated over related models such as COCOMO (Boehm, 1981).

[11]Also came up with an approach to optimize COCOMO –II model and further evaluated the two models on the TURKISH INDUSTRY data sets. The presented method has ability to deliver good estimation competencies even though it requires further enhancement. An assessment GA method for prediction the cost estimation of software using DLOC and the measure strength emerged from [28].

Simulated Annealing

Three[29] renowned researchers based their work on utilizing Simulated Annealing (SA) together with Bayesian Classifiers (BC) to guess the stability of software system. The authors' method proceeds the BC as input and acclimatizes it by utilizing SA. The method isobserved to be much superior to the outcome of good initial adept built using BC. [30]Emerged with effort software component estimation. Method. The method is validated against that of Sheta (2006) and the approach is observed to be powerful estimation model.

Ant Colony Optimization

Ant Colony Optimization (ACO) usage method is recommended by [31] for software error – estimation. The introduced model is called Ant Miner+ which utilizes graph execution of the classification rules. Every metric is a node. The rout which the ant takes is categorized to be the classification itself. For the reasons of assessment, three comprehensives open – sources data sets from NASA software projects were utilized. They are; KC1, PC4 and PC1. The method has the ability to compete with other methods such as C4.5, SVM, and logistic regression, in terms of instinctiveness and unambiguousness. [32]presented an adaptive method that takes already existing models as input and acclimatizes them to fresh invisible data. Ant Colony Optimization procedure is constructed for this reason. The author assessed the method on steadiness of classes in an Object – oriented system. The evaluation outcomes show the preeminence of the presented model over C4.5 and arbitrary predicting.

Particle Swarm Optimization

[33]Introduced a comprehensive work on Particle Swarm Optimization (PSO) for estimation of software strength using the KLoC metric. Fuzzy logic software model is also introduced. The models are evaluated on traditional methods such as Walston- Felix, Bailey – Basiliadn Doty on NASA's eighteen complete data sets. The outcomes reveal the preeminence of the two models being introduced. A multi – objective PSO for the strength of software estimation is presented by [34]. The inputs to the model are lines of cod size and the strength multiplier metrics. The models are validated on COCOMO model on two complete data sets, it was observed that, the introduced model gives better estimation, particularly on the initial data set with reduction error rate percentage.

Hybrid Methods

Three classic GA hybrid methods are presented by [11]. In the first phase, the authors hybridized GA with SA and in second phase with TS. In the third hybrid method, it combines GA, SA, and Tabu Search (TS) respectively. All the hybridized approaches are validated against C4.5 algorithm and a typical GA from Azar and that of Precup (2007) on complete outstanding data sets explaining the reliability of classes in an Object – Oriented Programming system. The hybridized methods outweigh other methods, hence, proving outstanding outcomes. But the hybrid methods need lot of implementation period. Again, the outcome of the hybrids are relatively multifaceted. A fusion of GA and Support Vector Machine (SVM) for inter – release error estimation utilizing the metrics from Chidamber and Kemerer (1994) is introduced by [35].

The principal goal of this GA is to locate appropriate setting of parameter. The method is validated against 6 well-known machine learning approaches. These include Naïve Bays, Logistic Regression, C4.5, Multi – Layer Perceptron, K – Nearest Neighbor (K- NN) and Random Forest. Tenfold cross assessment is used. The results

indicate that, the measurements for evaluation are accurate, precise, recall and F – measure. When, assessed on the jEdit PROMISE complete data set from (Promise), the fusion is observed to be very operative particularly for inter – release error estimation.[36]Recommended a comprehensive hybrid approach. The authors not only assessed the method on 6 machine learning methods, but also over variants of SVM. It is authenticated on numerous PROMISE data sets from (promise). Such as Log4j versions (1.0, 1.1)

The hybrid proves to be very operative even though it needs more execution period than related models. [37]Emerged with the application of GP together with Artificial Bee Colony (ABC). The experiments were carried out on complete NASA data sets KC1, PC1 and mushroom data set. The hybrid method is equated to neural gas, support vector machines as well as symbolic regression. The correctness on assessing is calculated applying 10 fold cross evaluation. The hybridized GP –ABC achieves outstanding outcomes on the mushroom data set outweighs other data sets by average value of 2.9%. An outstanding hybrid approach on Ant Colony Optimization (ACO) and Genetic Algorithm (GA) is presented by [38]. The aim is to optimize software cost estimation using the KLoC metric. Up to ten complete NASA data sets are utilized as yardsticks. The approaches is found to be much more operative than COCOMO model from (Boehm, 1981) with respect to the Magnitude of Relative Error (MRE).

Artificial Bee Colony

A multi –layer perceptron (MLP) neural network utilizing ABC for the guessing of software error is introduced by [11]. The authors' method is equated to MLP with back proliferation. They ended that, if correct parameters are establish to ABC, the neural network can be more successfully trained. Both methods are equated to: MLP trained utilizing ABC (MLP – ABC) against MLP trained utilizing back propagation (MLP – BP). The test is carried out on the three complete NASA data sets; CM1, KC1 and KC2. Testing the correctness and precision, it was found that, MLP –ABC outweighs MLP- BP by average value of 1.4% and 1.8%. Software algorithm to specifically optimize the prediction correctness of artificial neuralnetworks (ANN) is introduced by [39].

The authors trained the proposed approach (ANN) by utilizing swarm intelligence methods. These include; PSO, ACO, ABC as well as firefly. The principal goal is to achieve the best parameter for the propose ANN, that is the number of input neurons, number of hidden layers and hidden neuron, number of output neuron, weights etc. they equated ANN – PSO, ANN –ACO, ANN- ABC and ANN- firefly on complete data sets from NASA namely; Arc, Camel (1.0, 1.2, 1.4, and 1.6), Intercafe and Tomcat respectively. The authors found that, ANN –PSO is best method which had the best outcome in seven out of eight data sets. The most second classified approach is ANN- ABC obtaining best outcomes in three out of the eight data sets.

Parameters and C4.5 Input

C4.5 algorithm assents input as data set, an instance is given in table 3. Here, the table contains 14 complete data set made up of 10 cases, 4 attributes which include (LOC, DIT, LOCO and NOC) and a single classification label showing the reliability of software. However, the input data set must gratify the following conditions,

1. The data must be written the inform of vector of attribute values
2. The label must have initial definition and smartly outlined
3. The number of cases must be greater than the labels.

The data set is equally divided into two. One for testing and other for training. Here, the size of the training and testing is decided by the user. C4.5 trains data set and outputs a classifier. The enactment is always tested on training and testing data sets.

Table 5:- C4.5 inputs data sets.

Data case	DIT	LOC	LOCO	NOC	Reliability
1	5	86	86	2	reliability
2	5	81	91	1	reliability
3	3	84	78	2	unreliability
4	2	71	96	2	unreliability
5	2	70	80	0	unreliability
6	2	69	71	5	reliability
7	2	65	73	0	reliability
8	3	68	65	2	unreliability

9	5	72	92	2	unreliability
10	2	69	71	6	reliability
11	5	75	70	0	unreliability
12	3	72	91	0	unreliability
13	3	80	76	2	reliability
14	2	71	80	0	reliability

Entropy and Gain

For decision trees to be built and decide for the best one devoid of repeating through all the decision trees, C4.5 algorithm utilizes two approaches namely; entropy and information gain criterion. Here, the event which occurs with probability is denoted by P [M]. Now the information gain criterion is calculated from the equation below. However, to demonstrate this, for instance, a data set where all the data points have a particular class d. by denoting M to be the event of categorizing a data point as class d . Here, P [M] = 1. By equation below, information [M] = 0. Therefore, the information gain criterion is considered as the sum of ambiguity in the result of the event.

Information = $\log_2 \frac{1}{P[M]}$ (7)

The information gain criterion is linked to the entropy. Now let A denotes a data set of P positive classification labels and n negative classification labels. Here, the entropy can be calculated from the equation below. The entropy is actually dependent on the likelihood distribution of the classification labels. That is, if the entire data points in A belong to a single class, then, the data set is imbalanced, else it is unbiased.

The entropy shows the distribution of the data set; either balanced or not balanced.

Entropy [A] = $-\frac{\mathbf{p}}{\mathbf{P+n}} * \mathbf{log_2} \frac{\mathbf{p}}{\mathbf{p+n}} - \frac{\mathbf{n}}{\mathbf{p+n}} * \mathbf{log_2} \frac{\mathbf{n}}{\mathbf{p+n}}$ (8)

In the general form, the entropy can be expressed as follows where **Pt** denotes the probability of possessing classification label t

= entropy [A] = Pt * information [j]

= $\sum p_t * \log_2 \frac{1}{pt}$

= $\sum p_t * \mathbf{log_2 pt}$ (9)

Now let consider an instance where a case of data set possessing 14 cases belonging to similar class. Assuming p = 14 and n = 0 then its entropy is zero per the equation expressed below. Similarly, if P = 0 and n = 10. But $\mathbf{log_2\ 0}$ = is categorized as zero and 0 * $\mathbf{log_2\ 0}$ = 0

Entropy [A] = $-\frac{14}{14+0} * \log_2 \frac{14}{14+0} - \frac{0}{14+0} * \log_2 \frac{0}{14+0}$ (10)

= 1 $\log_2(1) - 0 = 0$

Now let also look at an instance of case of balanced data set [A] possessing 10 cases where p = 7and n = 7respectively. The nature of this entropy equals 1 per the following algorithm

Entropy [A] = $-\frac{\mathbf{7}}{\mathbf{7+7}} * \mathbf{log_2} \frac{\mathbf{7}}{\mathbf{7+7}} - \frac{\mathbf{7}}{\mathbf{7+7}} * \mathbf{log_2} \frac{\mathbf{7}}{\mathbf{7+7}}$

(11)

$= -\frac{1}{2}\log_2(0.5) - 0.5 \log_2(0.5)$

$= 2 * -(0.5) * \log_2(0.5)$

$= -(1) * -(1) = 1$

Assuming in the table above, the positive classification is reliable, where p = 5 and n = 9 respectively. Here, the Entropy can be calculated as follows

Entropy [A] = $-\frac{5}{9+5} + \log_2 \frac{5}{9+5} - \frac{5}{9+5} * \log_2 \frac{9}{9+5}$ (12)

$\approx -0.4 \log_2(0.4) - 0.6 \log_2(0.6)$

$\approx -0.4 * (-1.3) - (0.6) * (-0.7)$

$\approx 0.52 + 0.42 \approx 0.94$

It can vividly be seeing from here that, the entropy is very close to that of balanced data set.
Additionally, let a denotes an attributes which takes d discrete values where C4.5 splits the first data set into k subsets $S_{1,}\ldots,S_{k.}$ every subsets $S_t$possesses P_t positive labels N_t negative labels.
Therefore, the conditional entropy is calculated as follows where D | a is read as "D given a". It actually demonstrates quantity of bits information needed to categorize the example.
If the Entropy [D | a] = then the data is said to be flawlessly separated.

$$\text{Entropy } [D \mid a] = \sum \frac{p_t + n_t}{P+N} * \text{Entropy } (S_t) \qquad (13)$$

Prior to the testing of the attribute a, the conditional entropy is required to measure the information gain. Here, the information gain is represented by GI where C4.5 selects best information gain when splitting a complete data.
IG (a) = Entropy [D] – Entropy [D | a]

C4.5 Algorithm
The algorithm of C4.5 constructs classification decision tree based on divide and conquer "recursive algorithm".

IF thereareno cases inthe training set**THEN**
Create a leafnode and label it using some other knowledge source
ELSE
IF all casesinthetrainingsetare of the same category**THEN**

Create a leafnode and label it with the name of this category
ELSE
Select one attribute
Performa testbasedonthisattribute
Performa testbasedonthisattribute

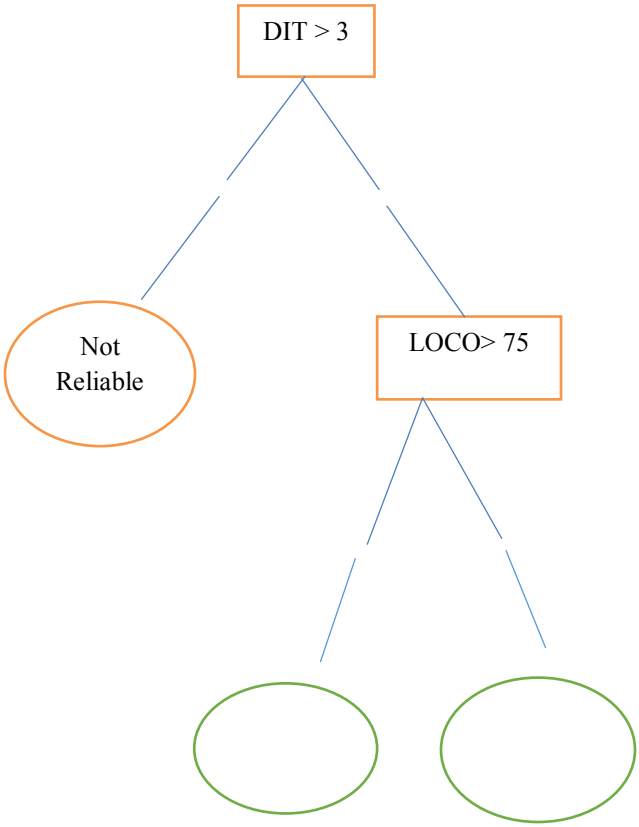
Divide thetrainingset into subsets, each associated with one possible value of thetest outcome
Repeat thealgorithmabove with each subset of the training set.

ENDIF
ENDIF

The principal purpose of C4.5 algorithm is to construct a decision tree with substantial prediction power. Preferably, the outcome decision tree is compact. An example has been given below.

Assuming, C4.5 is input a data set from the table 5 where the root of the tree can any attribute of DIT, LOC, LOCO or NOC. Here, utilizing the entropy and the gain criterion procedures, it can be inferred by C4.5 that, the "DIT > 3" has the outstanding procedures if it is selected as the root node. The data is then split per this assessment. Again, C4.5 calculate these measures assigningthis root and inferred that the previous test is whether LOCO is more than 75. Therefore, this assessment is demonstrated in the subsequent level of the decision tree. This approach is recurrent till the data can no longer be split into further subsets thereby obtaining the final decision tree

Figure 5:- DecisiontreewithDITasparentnode.



Artificial Bee Colony

Artificial Bee Colony is innovative swarm intelligence method based on collective beesfor scavenging behavior. The basic idea is that, these animals collaboratively share information to achieve optimum goal. It was initiated by [40]. The author addressed that, Swarm intelligence depends on the intelligent behavior of swarms such as ants scavenging behavior method, Ant Colony Optimization (ACO).

Common Bees in Nature

A group of bees can outspread over a long distance, approximately (up to 14km or 15km) in several ways to exploit a huge number of sources of good food. The source of the food is apparently denoted by a flower patch. Source of good food is one that gratifies the conditions outlined below:

1. The targeted food source must be near to the nest
2. The targeted food source must contain lot of energy
3. The energy of food source is easily haul out

Numerous kinds of bees do live in one colony, such as the worker bees, can also be classified as forager bees and are the ones to find and collect good food sources. However, the worker bees can be categorized into employed or unemployed forager bees. The employed forager bees essentially profit from a good food source and haul out good energy from it. They actually send signals of information concerning this source when returning to the hive. This information includes

1. The actual distance from the hive to the source of the food
2. The way to follow from the hive to the source of the food utilizing sun as location reference
3. Finally, the lucrativeness of the food source; this means, the quantity of the haul out nectar from the food source

It is pertinent to note that, the employed bee has a calculated limit of trials it goes in finding improved source of food. If the trial limit is reached, it will automatically leave the food source and immediately becomes a common unemployed forager bee, specifically a lookout.

The unemployed forager bees are ones that are to locate a source of good food and there are two types. They are scouts and onlookers. The scouts tirelessly continue to reconnoiter locations arbitrarily around the nest to discover a good food source whilst the onlookers patiently stay in the nest to wait for the employed forager bees to return and share information about the source of thefood.

When the onlooker bees receive the information about the food source from the employed forager bees, they then decide which food source to go depending on the lucrativeness of the food source. It can be noted here that; the exploitation of a good food source is done by the employed forager bees whilst the exploration of food source is done by unemployed forager bees.

Figure 6:- the diagram below vividly describes the aforementioned information.

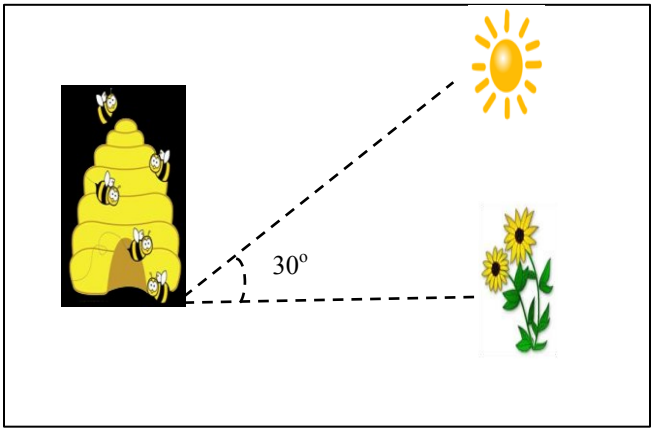


Table 5:- Types of common bees in a bee colony.

Bee	Task	Quantity & sex
Scouts & onlookers	Find good food sources	
Queen	Lay eggs	One female
Drones	Mate with queen bee	Numerous male bees
Workers: employed & unemployed forager bees	Profit from good food source	Hundreds & thousands of female bees

ProposedABC Algorithm

The algorithm of Artificial Bee Colony is enthused from the scavenging behavior of bees in nature. Every food source equals the solution of the problem in the search area. Basically, the lucrativeness of a food source is determined by the ABC algorithm. The algorithm computes the predefined objective function. Actually, ABC begins by arbitrary initialization of the solution search area. It does perform in three stages.

1. Stage one comprises of sending the working bees:the nectar amount denotes the excellence solution; that is the objective function
2. Stage two comprises of sendingtheobserverbees, observers' bees are employed depending on a probability-based selection process. Thus, a solution thatis exceedingly lucrative has a complexlikelihoodofbeing selected by observer bees.
3. Stage three comprises of sending the lookout bees.

Lookout bees use random search to locate good solutions. According to [40] they are categorized by low search costs and a low average in source of food excellence. The lookout bee locates new source of food fast, these sources are not essentially lucrative. A working bee is changed into lookout bee, if it cannot locate a better solution within the specified "limit" parameter, it will automatically leave the food source and become a lookout bee. The ABC algorithm saves the best solution found in the memory after the completion of all the three stages. The process continues iteratively until discontinue condition is met.

```
METHOD: Training Data SettrainDS, Testing Data Set testDS){

                initialize( ) ;
Repeat
sendEmployedBees();
calculateProbabilitiesOfSendingOnlookers ();
sendOnlookerBees ();
sendScoutBee ( );
memorizeBestFoodSourceFoundSoFar( );
UNTIL                (stoppingCriterionIsMet)
return bestFoodSourceFound;
}
```

The artificial bee colony comprises of numerous parameters such as swarm size, number of onlooker bees, and number of employed bees, limit and number of scout bees. The initial size of the bee colony is the swarm size. The bee colony has two equal parts. The first part embroils of employed scavenger bees and second scout bees. In the real sense, the number of onlooker bees as well as number of employed bees are the first number of onlooker and employed bees. This implies that, the greater the number of onlookers, employed and scout bees, the greater the size of the swarm. The number of employed as well as onlooker bees are often set to have equivalent number.

The first number of scouts is number of scout bees.

This number however, depicts the number of bees that can reconnoiter the search area. The greater the number, the greater the exploration area of fresh solution is. This parameter however, should be wisely selected, because if it is too high, the algorithm converges to guessing search. It always advisable to set it to one. Limit denotes the actual number of candidate solutions which the onlooker bee can explore devoid of enhancing the present solution. The onlooker bee always tries to use trivial alterations to the present solution in an effort of enhancing its objective function. Once the solution doesn't seem to enhance after a given number of times, the solution is abandoned. Here, the bee has a chance of becoming a scout so as to explore the search area again.

Table 6:- proposed parameters of ABC.

Number of the parameter	**explanation**
Number of scouts	First number of scouts
Number of employed bees	First number of employed bees
Number of onlooker bees	First number of onlooker bees
limit	The number of candidate solutions that a bee can search devoid of enhancing the present solution
Swarm size	First size of the bee colony

Here, we took one scout bee into consideration. The number of employed bees as well as number of onlooker bees is each equivalent to the number of rule sets that is accessible. A typical instance, if 6 rule sets are available, then 6 employed bees and 6 onlooker bees will be available as well. The summation of swarm size in this instance is 13 that is (6 employed bees + 6 onlooker bees + 1 scout bee). It must be noted that, the number of rule set is firm, so that each cycle does not swing. We set ending criterion to a static number of cycles. Objective function is the equation for the proposed ABC.

$$f(\text{Ruleset}) = w\alpha * \text{Accuracy} + w\,j * \text{Balanced accuracy} \qquad \textbf{(Eq14)}$$

where**wα** denotes weights of Accuracy and Balanced Accuracy. We present the general form of source code for ABC below

```
Method ABC; (Training Data Set train DS, Testing Data SettestDS){
```

```
            initialize ( ) ;
            memorize Best Rule Set Found So Far( );
intcycle = 1;
Repeat
Send Employed Bees( );
calculate Probability Of Sending Onlooker();
            send Onlooker Bees ( );
            send Scout Bee ( );
            memorize Best Rule Set Found So Far( );
             cycle = cycle +1;
              UNTIL(cycle = maximum_cycles)
               return best Rule Set Found;}
```

The code begins by utilizing the algorithm C4.5 of machine learning to acquire numerous rule sets. For every rule set, wecompute its accuracy (Eq. 1), balanced accuracy (Eq.3) as well as precision (Eq.6) on data set training. We further compute every rule set's objective function on the training set. For every metric, the set of cut point average value is also calculated. The metric values that affect the classification label is classified as cut point. For this reason, the data set is actually sorted by the metric.

We also present the initialization source code for employed bees below

```
Method; initialize ( )
{
 getC4.5Generated Rule Sets( );
FOR EACHRuleSetrs DO{
Calculate Performance Measures (rs , trainDS);

Calculate Objective Funtion (rs, trainDS);

}
Get Cut points Per Metric(trainDS);
}
```

At this juncture, the employed bee is sent to each rule set. The pseudo code is displayed below respectively.

```
Method; send Employe dBee (Rule Set rs) {
FOR EACHmetricinrsDO{
FOR EACHrule in rsDO{
FOR EACH condition DO {
mr = change Operator Sign(rs);
mr = change Condition Value(rs);
  calculate Objective Function ( mr , train DS, test
DS );
IF ( f(mr)>f(rs) ) THEN{
keep mr and delete rs;
return;
}ELSE
{
keep rsanddeletemr;


      }
   }

  }

}
```

Iteratively, the employed bee goes over every metric in this chosen rule set and locate the available condition in which it finds itself. A change of these conditions take place here by the bee. The changes can be in two folds. Either the bee modifies the value of the condition or the operator of the condition. In the previous case, there is random selection of value from the list of cut point values of the metric in question. In the last example, the bee actually, selects an operator from the giving set of allowed operators randomly. A vivid example, if the condition is defined as "NOC > 9" which invariably could become "NOC≤ 9". The changed rule set's objective function is calculated on the training data. If the rule set is seen to be better, it is kept, else the modification is vetoed.

The prelims of source code for onlooker stage is presented below.

```
Method; calculateProbabilities (Rule Setr1,… , RuleSet rk) {
FOR EACHRuleSetrsDO{
getObjectiveFunction (rs);
}
distributeOnlookersByProbabilityBasedSelection( );
}
```

This stage commences by computing probabilities of sending the onlookers. Specifying the rule sets, the algorithm initially calculates for every objective function. Then, the rule sets are commanded decreasingly depending on their objective function. The rule set possessing the tiniest objective function is detached from search area. If the rule set is more than one with equivalent smallest value of the objective function, all the underlying rule sets are automatically detached. For instance, if a rule sets are detached, then extra n onlookers are immediately sent to the highest objective function of the rule set.

Source Code to Send Onlooker Bees is also presented in the box below

```
Method; sendOnlookerBee(RuleSetrs ) {
inttrial = 0;
FOR i=1 TO limitDO{
FOR EACHrule r inrsDO{
mr = rs;
FOR EACHcondition in r DO{
mr = changeCondition ( mr);

mr =changeRuleClassLabel( mr);
 }
```

```
}
changeRuleSetClassLabel( ) ;

calculateObjectiveFunction ( mr , trainDS, testDS);
calculateObjectiveFunction ( mr , trainDS, testDS);
IF ( O(mr)> O (rs)  ) THEN{
keep mr;
return;
}
ELSE{
keep rs;
rs.setTrial (trial ++);
                                        }
                                }


                        }
```

Scout bee stage

Scout bee is the last operative stage of the ABC algorithm. This stage is the main of the algorithm's exploration stage. Here, the rule set with maximum number of trials is detached. The scout bee then generates a fresh rule set with arbitrary number of rules between 1and 5. Utilizing these precise intervals, better rule sets are acquired. Additionally, these intervals are experimentally selected for thispurposebecause they retain the readability and easiness of a rule set.

```
Method;sendScoutBee ( ){
scoutIndex= getScoutBeeIndex( );
intnumberOfRules = random(1,5);
intnumberOfConditions= random(1,5);
scout= createRuleSet(numberOfRules,   numberOfConditions,scoutIndex);
returnscout;
}
```

All the respective stages of ABC are recurrent for a certain static number of cycles. Upon completing all three respective cycles, the rule set with the outstanding objective function on the training data is reverted.

Results and Discussions:-

Stab One

This data set clearly explains the steadiness of classes in typical Object – Oriented Programming systems. The metrices are assembled into two, comprising three or four metrices. The Stress metric is always added to every group. All the probable groupings are established which end up creating 15 subsets of formed groups. All metric subsets together with 11 complete data systems create data sets. (Table 10). Using C4.5 helps in building decision trees. All the metric with continuous classifier or with a fault greater than 10% are automatically detached. About forty decision trees are arbitrarily chosen from the rest. The metrics are changed to rule sets. All the facsimile rule sets are detached remaining 30 rule sets to deal with.

Stab Two

Another comprehensive data set that explains the steadiness of class in the environment of Object – Oriented Programming is STAB 2. It has largely been utilized in the works of many scholars such as Azar etcetera. The metrics utilized in STAB 2 are well defined in (table 11). These metrics were present by [41]. They are haul out from the systems presented in (table 12) utilizing ACCESS tool system as well as Discover environment ©. Likewise STAB1, complete fifteen subsets of metrics are built. Utilizing these metrics together with nine software

systems (13). Decision trees with a continuous classifier with a fault greater than 10% are detached. Here, decision trees are then changed to rule sets. All the facsimile rule sets are detached which end up acquiring 20 rule sets.

Table 7:-Tabular form of software reliability metrics.

Giving name	explanation
Size of intricacy metrics	
NOM	numberof procedures
NPA	numberofpublicattributes
NPPM	numberofpublicandprotectedmethodsina class
MCC	McCabe'scomplexityweightedmethodsperclass
WMC_LOC	LOCweightedmethodsperclass
Metric of Inheritance	
NOC	numberofchildren
NOP	numberofparents
DIT	depthofinheritance
MDS	messagedomainsize
CHM	classhierarchymetric
Coupling Metrics	
OCMAIC	otherclassmethodattributeimportcoupling
CUB	numberofclassesusedbya class
OMAEC	otherclassmethodattributeexportcoupling
Cohesion Metrics	
COH	cohesion
LCOMB	lackofcohesionmethods
COM	cohesionmetric
COMI	cohesionmetricinverse
Classification Label	
Steadiness	{0=stable,1=unstable}
Stress Metric	
Stress	stressappliedtotheclass

Data Sets Used

Table 8:- data sets explanation linked to software stability.

Data set	Over all attributes	Over all data instances	stable	unstable
Stab One	21	2920	440(15%)	2482(85%)
Stab Two	24	690	236 (35%)	456(67%)

Table 9:- data sets explanation linked to software faults.

Data set	Over all attributes	Over all data instances	stable	unstable
CM1	23	499	49(9.49%)	448(90.16%)

First Experiment

In the first experiment. STAB two was utilized aswell as objective function.

f (N) = [Correctness]

The technique was validated against mainstream classifier C4.5, GA from [42]. In the process, the outstanding rule set noticed by ABC, the proposed method outweighs the majority classifier by average percentages of 8.6%, 6.66% and 14.7% on the chosen training correctness, training balanced correctness, testing correctness and testing balanced correctness correspondingly. The technique further, enhances on C4.5 by 7.15% and 8.26% on training correctness and training balanced correctness. Additionally, C4.5 is outweighed by the proposed technique by 5.63% and 8.23% on testing correctness and testing balanced correctness correspondingly.

Table 10:- displaying first experiment on STAB two, correctness and balanced correctness on complete training data sets. Inside the parentheses are the values of standard deviation.

heuristic	Correctness, Training Standard deviation	Balanced Correctness Standard deviation
majority	67.1%(0.81)	50%(0)
ABC	79.7%(1.03)	69.9%(1.4)
SA-TS-GA	77.28%(0.41)	68.17%(1.13)
TS-GA	74.08%(0.58)	65.72%(0.99)
SA-GA	76.01%(0.54)	69.20%(1.08)
C4.5	67.56%(0.7)	57.43%(0.5)
GA	74.5% (1)	65%(3)

Table 11:- displaying first experiment on STAB two, correctness and balanced correctness on testing data sets. Inside the parentheses are the values of standard deviation.

heuristic	Correctness Testing Standard deviation	Balanced Correctness Standard deviation
C4.5	68.97%(5.57)	58.47%(3.50)
Majority	66.97%(5.57)	58.47%(3.50)
GA	70%(6)	61.5%(5)
SA-GA	72.58%(5.04)	64.94%(4.94)
TS-GA	70.52%(5.53)	62.05%(3.93)
SA-TS-GA	70.59%(5.38)	63.33%(4.50)
ABC	73.6%(5.34)	64.7%(4.83)

Figure 7:- Displaying result of first experiment on STAB two, accuracy of every heuristic on training as well as testing data.

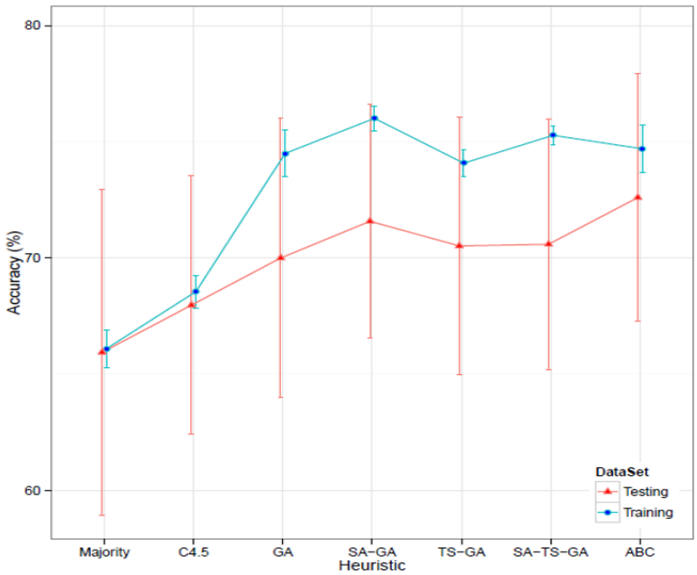
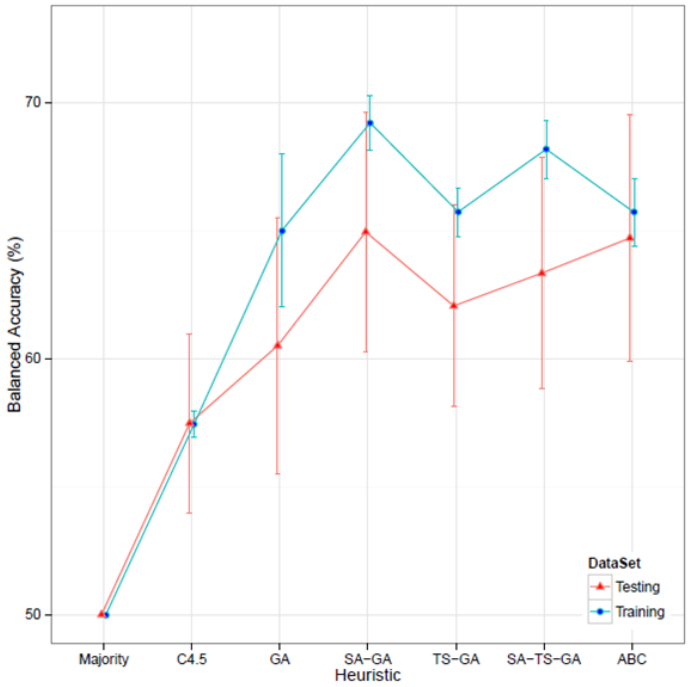


Figure 8:- displaying result of first experiment on STAB two balanced accuracy of every heuristic on training as well as testing data set.



Second Experiment

The second experiment isdone on STAB one. The researcher actively utilized the objective function to give equivalent weights to correctness and balanced correctness

f(N) $\mathbf{1/5}$*(correctness + Balanced correctness).

The technique was validated over majority classifier, C4.5, GA, a work from Azar and Precup (2007) and further to three adaptive hybrids namely; SA – GA, TS – GA and SA- TS- GA. A work presented by [42]. Because STAB one is very imbalanced data set, we added the balanced correctness in the objective function, presenting both correctness and balanced correctness equivalent weight.

The proposed method improves on majority classifier by 1.74% and 2.64% on both the training and testing correctness correspondingly. However, on the training and testing balanced correctness, the technique is improved by 29.94% and 27.6% correspondingly. The presented method improves on C4.5 by 19.27% and 22. 72% on training correctness and training balanced correctness.

In conclusion, the proposed ABC method performs better than others on the training and testing correctness, training and testing balanced correctness. Because STAB one found to have imbalanced data set, balanced correctness is considered to be quality criterion. The enhancement of the presented technique on the testing balanced correctness has a very high average point of 20.7% which is found to be very encouraging.

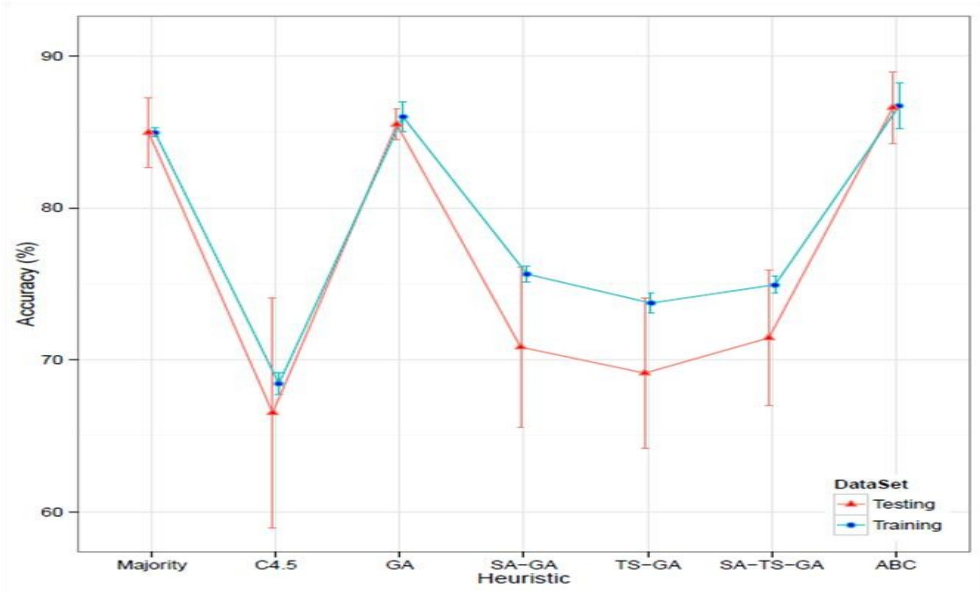
Table 12:- displays the results of second experiment on STAB one, correctness and balanced correctness on training data sets. Inside the parentheses are the values of standard deviation.

Heuristic	Correctness training Standard deviation	Balanced correctness Standard deviation
ABC	86.71%(1.5)	79.94%(0.83)
SA-TS-GA	75.95%(0.55)	69.71%(0.74)
TS-GA	73.74%(0.66)	67.71%(0.74)
SA-GA	76.65%(0.53)	67.31%(1.24)
GA	86%(1)	60.50%(1)
C4.5	69.44%(0.7)	58.22%(1.12)
Majority	84.97%(0.3)	50%(0)

Table 13:- displays the results of second experiment on STAB one, correctness and balanced correctness on testing data sets. Inside the parentheses are the values of standard deviation.

Heuristic	Correctness testing standard deviation	Balanced correctness testing Standard deviation
C4.5	68.52%(7.56)	57.7%(3.34)
Majority	86.97%(.23)	50(0)
GA	87.5%(1)	59(4)
SA-GA	73.84%(5.3)	65.17%(5.27)
TS-GA	69.13%(4.92)	63.52%(3.82)
SA-TS-GA	71.45%(4.47)	66.88%(4.78)
ABC	87.61%(2.36)	79.6%(3.48)

Figure 9:-Displaying the graphical representation of second experimentonSTAB one.Correctness of everyheuristicontrainingandtestingdata set.



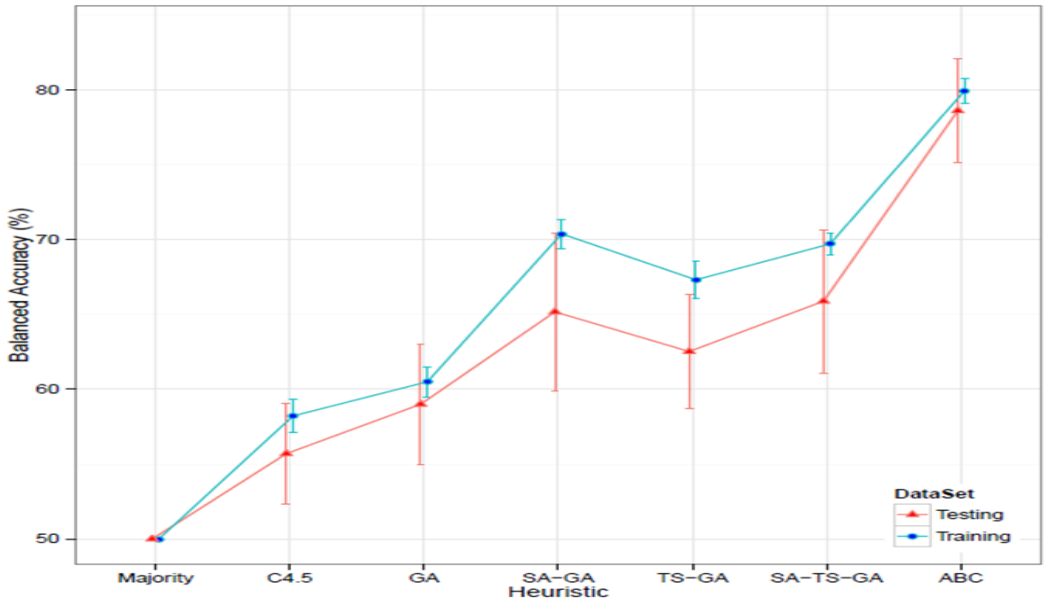


Figure 10:- displaying the graphical representation of second experiment on STAB one of balanced correctness of every heuristic on training and testing data, respectively.

Third experiment

In the process of this experiment, STAB one is again utilized but the objective function used is expressed below **f (N) = Balanced Correctness**. The proposed technique is validated againstmajorityclassifier,C4.5,GA.Aswellasthreehybridsnamely**SA–GA, TS-GA**and **SA – TS – GA**. Because STAB one has imbalanced date set, majority classifier would reach a great correctness devoid of being able to categorize the minority classification label. Because of this motive, balanced correctness was integrated in the objective function.

For majority classifier, it was confirmed same as training and testing correctness hence escalates on the balanced correctness by 30.87%and29.55%ontrainingandtesting.Further,theapproachimprovedmassivelyonC4.5by 6.22%, 14.39%, 5.9% and 14.56% on the training correctness, training balanced correctness, testing correctness and testing balanced correctness, respectively.

It was found that, GA outweighs the proposed approach by 3.26% and 3.28% on both training and testing correctness. In return, the proposed ABC outweighs GA by 21.39% and 20.54% respectively on training and testing balanced correctness. The presented algorithm outweighs SA – GA by 4.85%, 1.72%, 4.16% and 3.09% on the training correctness, training balanced correctness, testing correctness and testing balanced correctness, respectively.

Table 13:- Displays the results of third experiment on STAB one, correctness and balanced correctness on testing data sets. Inside the parentheses are the values of standard deviation.

Heuristic	Correctness testing standard deviation	Balanced correctness testing standard deviation
C4.5	79.32%(1.98)	66.98%(4.26)
Majority	85.97%(2.3)	50%(0)
GA	85.5%(1)	59%(4)
SA-GA	80.06%(2.16)	77.45%(2.17)
TS-GA	78.38%(1.79)	72.86%(5.03)
ABC	83.22%(2.77)	79.54%(2.94)
SA-TS-GA	77.90%(2.38)	75.86%(3.23)

Figure 11:- Shows the graphical representation of third experiment on STAB one of every heuristic on training and testing data respectively.

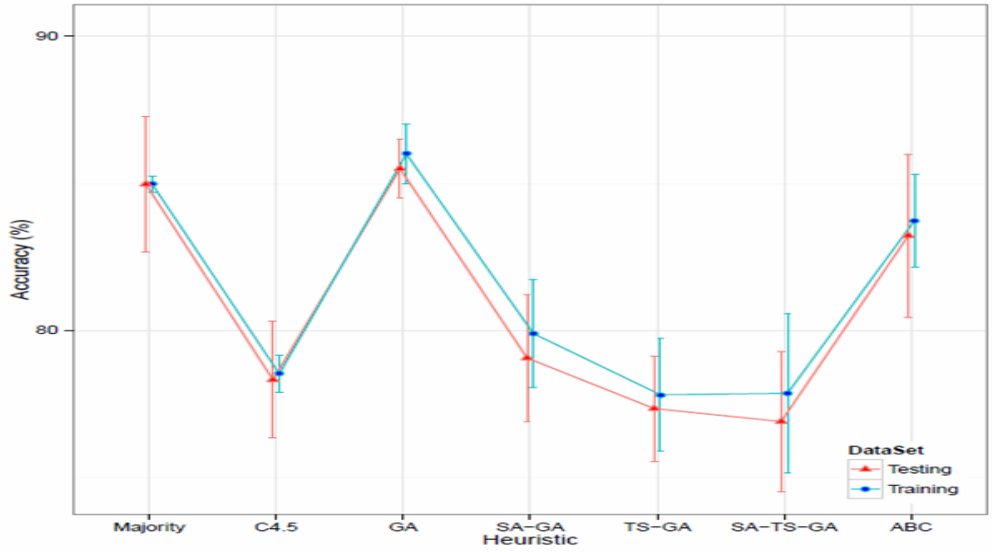
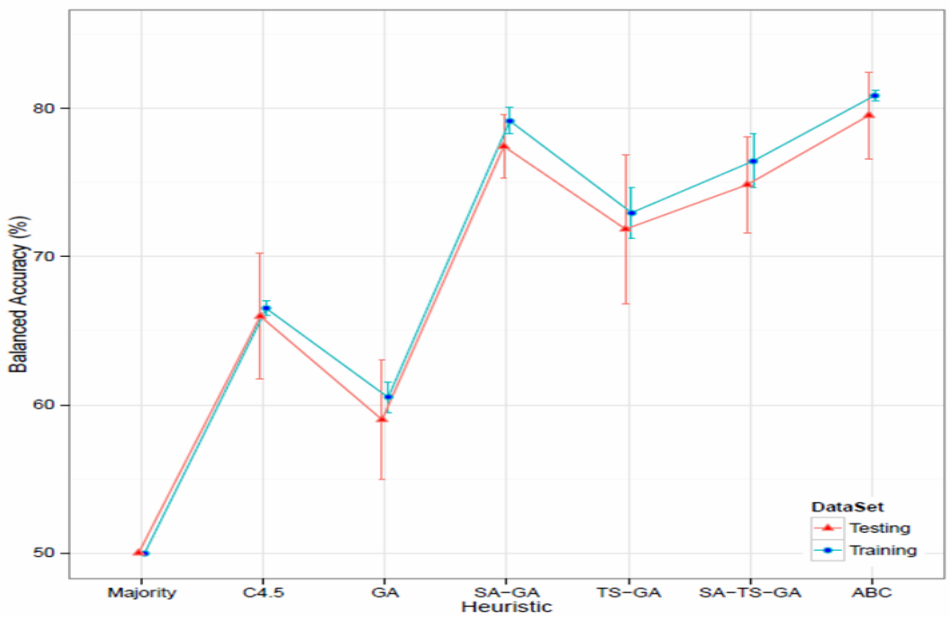


Table 14:- Displays the results of third experiment on STAB one, correctness and balanced correctness on testing data sets. Inside the parentheses are the values of standard deviation.

Heuristic	Correctness testing Standard deviation	Balanced correctness standard deviation

ABC	82.22%(2.77)	80.54(2.94)
Majority	85.97%(2.3)	52%(0)
C4.5	79.32%(1.98)	65.98%(4.26)
GA	86.5%(1)	58%(4)
SA-GA	79.06%(2.16)	77.45%(2.17)
TS-GA	78.34%(1.79)	72.86%(5.03)
SA-TS-GA	76.90%(2.38)	75.86%(3.23)

Figure 12:- Shows thegraphicalrepresentationofthirdexperimentonSTABonebalanced correctness of every heuristic on training and testing datarespectively.



Fourth Experiment

Here, CM1 is largely used and set the correctness as the chosen objective function to validate the proposed method of neural network multi- layer perceptron MLP. MLP with Back Propagation (MLP –BP) as well as MLP combine with ABC (MLP –ABC). The approach is further validated over majority classifier, C4.5 together with three hybrids namely SA – GA, TS- GA and SA – TS – GA. The work on CM1 prove to have much concentration on the precision measurement model since the data is linked to faults in safety critical software. It is however essential to categorize the faulty modules from non – faults ones.

Hence, the precision of the proposed model is well presented. The measurements on the training and testing data sets is well presented in the tables 22 and 23 whereas figures 11 -13 show the graphical representation of the measurement. MLP – BP and MLP – ABC are not part of the table and figures because good number of values were not deliberated therefore the researcher deliberated only those which were presented. On the training correctness, balanced correctness and precision, the proposed model advances by 2.79%, 15.34% and 78.83% correspondingly. The proposed ABC model acquires similar testing correctness as that of the majority classifier, whereas it escalates by 11.33% and 36.67% on the testing balanced correctness and precision correspondingly. It was found that, the proposed model performs weakly on C4.5. For that of, SA – GA, TS – GA and SA – TS – GA. A minimumescalationinthetrainingcorrectnessiswitnessedattheoutlayofamassivedepreciation inthetestingcorrectnessandtestingbalancedcorrectness.Here, chosendatasetislinkedtosafety critical software so high balanced correctness wasfocused.

Table 15:- Displaying the results of fourth experiment on CM1, correctness, balanced correctness as well as precision on training data sets. Inside the parentheses are the values of standard deviation.

Heuristic	Correctness training standard deviation	Balanced correctness Standard deviation	Precision, Training

			Standard deviation
SA-TS-GA	94.9%(0.79)	66.65%(5.13)	93.61(6.43)
ABC	94.95%(0.24)	65.34%(7.1)	79.83(15)
TS-GA	92.16%(0.53)	62.15%(3.88)	93.57(7.8)
Majority	93.16%(0.12)	50%(0)	1(0)
SA-GA	93.17%(0.5)	66.2%(2.4)	95.3(5.47)
C4.5	91.95%(0.24)	65.34%(7.1)	78.83 (15)

Table 16:- Displaying the results of fourth experiment on CM1, correctness, balanced correctness as well as precision on data sets. Inside the parentheses are the values of standard deviation.

Heuristic	Correctness Standard deviation	Balanced correctness testing standard deviation	Precision, training, standard deviation
C4.5	91.17%(2.17)	61.33%(10.42)	35.67(35.5)
Majority	91.17%(1.08)	52%(0)	1(0)
SA-GA	88.56(9.2)	57.08%(7.7)	40(46.6)
TS-GA	90.16%(2.1)	51.19%(3.85)	8.3(18)
SA-TS-GA	91.15%(2.87)	55.94%(8.7)	42.5(50.07)
ABC	91.17%(2.17)	60.33%(10.42)	43.67(35.5)

Figure 13:- Demonstrating the graphical representation of fourth experiment on CMI correctness of every heuristic on training and testing data respectively.

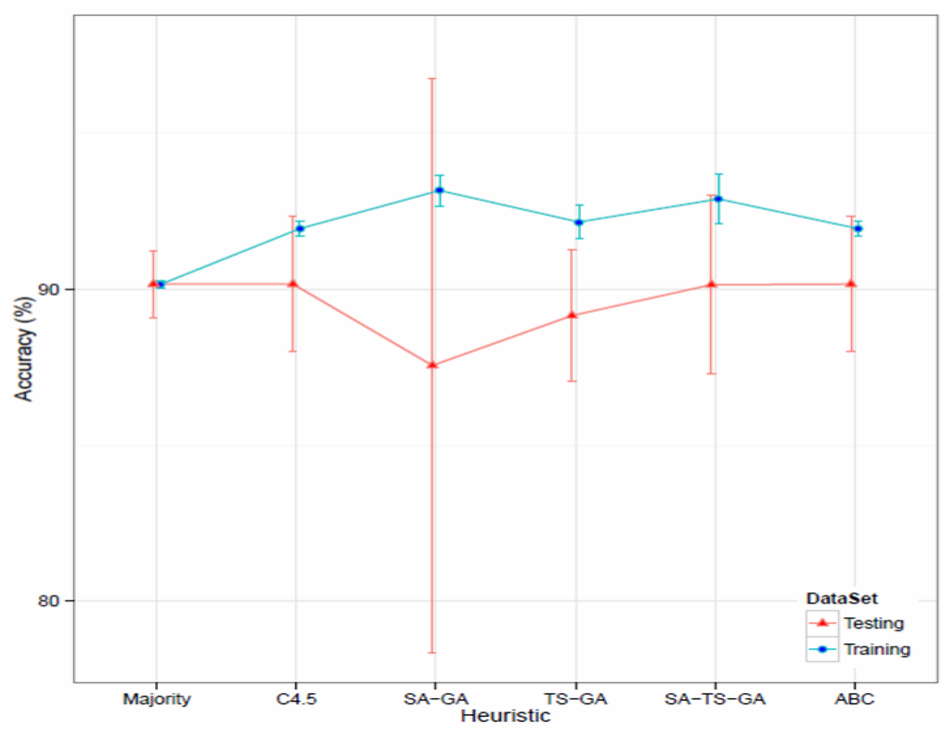


Figure 14:-Demonstrating the graphical representation of fourth experiment on CMI balanced correctness of every heuristic on training and testing data set respectively.

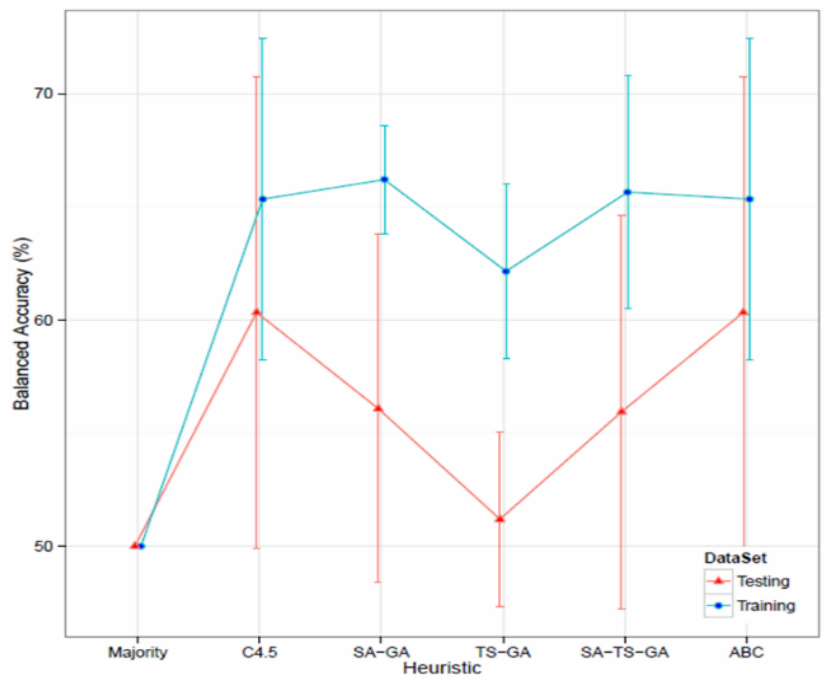


Figure 16:-Demonstrating the graphical representation of fourth experiment on CMI precision of every heuristic on training and testing data set respectively.

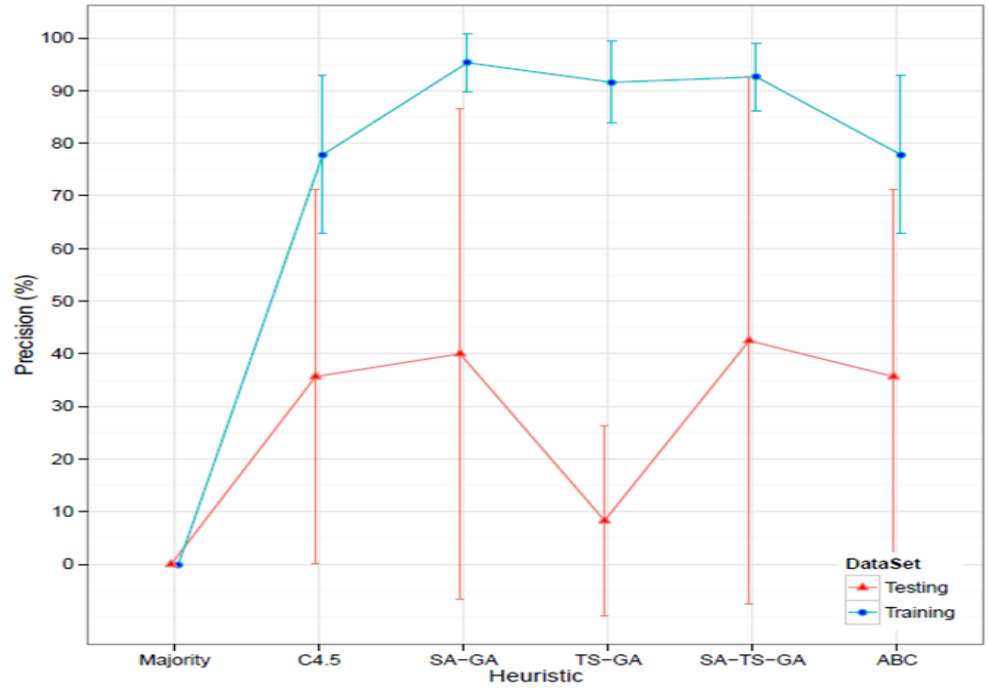


Table 17:- The table below displays the results of experiment 4 on CM1. This include the average values of rules per rule set as well as average value of conditions per rule. The value of standard deviation is presented in the digressions.

Heuristic	Average number of rules per rule set	Average number of conditions per rule (standard deviation)

	(standard deviation)	
C4.5	4(0)	2.2(0)
SA-GA	9.2(2)	12.55(2.7)
TS-GA	6.9(1.8)	17.9(3.4)
SA-TS-GA	8(2.3)	15.85(4.5)
ABC	2.5(0)	(0)

Results Summary:-

Table 18. This table shows the summary of the correctness calculated on the training data and the improvements on C4.5. Standard deviations are presented in digressions with all values in percentages.

	Experiment 1 on STAB 2		**Experiment 2 on STAB 1**		**Experiment 3 on STAB 1**		**Experiment 4 on CM1**		**Average**
	Value	Improvement	Value	Improvement	Value	Improvement	Value	Improvement	
Majority	66.1 (0.81)		84.97 (0.3)		84.97 (0.26)		90.16 (0.12)		81.55 (0.37)
C4.5	68.55 (0.7)		68.44 (0.7)		78.52 (0.63)		91.95 (0.24)		76.87 (0.57)
GA	74.5 (1)	+5.95	86 (1)	+ 17.56	86 (1)	+ 7. 48	N/A	N/A	61.63 (1)
GA – SA	76.01 (0.54)	+ 7.46	75. 65 (0.53)	+7.21	79.89 (1.84)	+1.37	93. 172 (0.5)	+1.222	81.18 (0.85)
TS – GA	74.08 (0.56)	+5.53	73.74 (0.66)	+5.3	77.81 (1.91)	-0.71	92.16 (0.53)	+0.21	79.45 (0.92)
SA – TS – GA	75.28 (0.41)	+6.73	74.95 (0.55)	+6.51	77.86 (2.71)	-0.66	92.9 (0.79)	+0.65	80.25 (1.12)
A B C	74.7	+6.15	86.71	+83.74	83.74	+5.22	91.95	+0	84.28

	(1.02)	(1.5)	(1.58)	(0.24)	(1.09)

Table 19:- This table shows the summary of correctness calculated on the testing data and the improvements on C4.5. Standard deviations are presented in digressions with all values in percentages.

Heuristic	Experiment 1 on STAB 2		Experiment 2 on STAB 1		Experiment 3 on STAB 1		Experiment 4 on CM1		Average
	Value	Improvement	Value	Improvement	Value	Improvement	Value	Improvement	
Majority	65.94 (7)		84.97 (2.3)		84.97 (2.3)		90.17 (1.08)		81.51 (3.17)
C4.5	67.97 (5.57)		66.52 (7.56)		78.32 (1.98)		90.17 (2.17)		75.75 (17.28)
GA	70 (6)	+2.03	85.5 (1)	+18.98	85.5 (1)	+7.18	N/A	N/A	80.3 (2.67)[8]
GA - SA	71.58 (5.04)	+3.61	70.84 (5.3)	+4.32	79.06 (2.16)	+0.74	87.56 (9.2)	-2.61	77.26 (5.43)
TS - GA	70.52 (5.53)	+2.55	69.13 (4.92)	+2.61	77.34 (1.79)	-0.98	89.16 (2.1)	-1.01	76.54 (3.59)
SA – TS - GA	70.59 (5.38)	+2.62	71.45 (4.47)	+7.93	76.90 (2.38)	-1.42	90.15 (2.87)	-0.02	69.77 (3.78)
ABC	72.6 (5.34)	+4.63	86.61 (2.36)	+20.09	83.22 (2.77)	+4.9	90.17 (2.17)	+0	83.15 (3.16)

Table 20:- This table shows the summary of the balanced correctness calculated on the training data and the improvements on C4.5. Standard deviations are presented in digressions with all values in percentages.

	Experiment 1 on STAB 2		Experiment 2 on STAB 1		Experiment 3 on STAB 1		Experiment 4 on CM1		Average
	Value	Improvement	Value	Improvement	Value	Improvement	Value	Improvement	
Majority	50 (0)		50 (0)		50 (0)		50 (0)		50 (0)
C4.5	57.43 (0.5)		58.22 (1.12)		66.50 (0.5)		65.34 (7.1)		61.88 (2.3)
GA	65 (3)	+7.57	60.50 (1)	+2.28	60.50 (1)	-6	N/A	N/A	62 (1.67)[9]
GA - SA	69.20 (1.08)	+11.77	70.38 (0.97)	+12.17	79.17 (0.89)	+12.67	66.2 (2.4)	+0.86	71.24 (1.34)
TS - GA	65.72 (0.95)	+8.29	67.31 (1.24)	+9.09	72.97 (1.72)	+6.47	62.15 (3.88)	-3.19	67.04 (1.95)
SA – TS - GA	68.17 (1.13)	+10.74	69.71 (0.74)	+11.49	79.46 (1.82)	+9.96	65.65 (5.13)	0.31	70 (2.21)
ABC	65.7 (1.3)	+8.27	79.94 (0.83)	+21.72	80.89 (0.36)	+14.39	64.34 (7.1)	+0	72. 97 (2.4)

Table 21:-This table shows the summary of the balanced correctness calculated on the testing data and the improvements on C4.5. Standard deviations are presented in digressions with all values in percentages.

	Experiment 1 on STAB 2	Experiment 2 on STAB 1	Experiment 3 on STAB 1	Experiment 4 on CM1	Average
Majority	50 (0)	50 (0)	50 (0)	50 (0)	50 (0)
C4.5	58.47 (3.5)	56.7 (3.35)	66.98 (4.25)	60.34 (10.42)	58.78 (5.38)
GA	60.5 +4.03 (5)	59 +3.3 (4)	59 -6.93 (4)	N/A N/A	60.5 (4.33)[10]
GA - SA	65.94 +7.47 (4.67)	65.17 +9.47 (5.27)	77.45 +11.47 (2.17)	56.08 -4.25 (7.7)	65.66 (4.96)
TS - GA	62.05 +4.58 (3.92)	62.52 +6.82 (3.82)	71.86 +5.88 (5.03)	51.19 -9.14 (3.85)	61.91 (4.16)
SA – TS - GA	63.33 +5.86 (4.50)	65.88 +10.18 (4.78)	74.86 +8.88 (3.23)	55.94 -4.39 (8.7)	65 (5.3)
ABC	64.7 +7.23 (4.83)	78.6 +22.9 (3.48)	79.54 +13.56 (2.94)	60.33 +0 (10.42)	70.79 (5.42)

Conclusion Remarks:-

In this paper, ABC as a method to optimize softwarereliability models has been researched.Systemreliability has beenconsideredas a caseinthispaper. The ABC algorithm has been proposed.The proposed algorithm mainly takes numerous rule sets as input and produces only one rule set from them. The results obtained show the efficacy of the methods proposed by improving the performance measures devoid of obscuring the size of the rule set. The easiness

of the rule sets generated makes it simple for human specialists to read, interpret and use as guidelines for software construction in future.

More importantly, the algorithm proposed has been user friendly on data sets with numerous classification labels, and data with different value types. To extend this work, further studies need to be conducted on the relationship between this approach and the nature of the datasets.Forinstance, howcorrectnesswouldchangewiththesizeofthedatasets,thenumberof metrics, which metrics are mostly used in the classification process, etc. Another interesting future work would be to test the algorithm on data sets describing other software quality characteristics.

References:-

1. Babayigit, B. and R. Ozdemir. A modified artificial bee colony algorithm for numerical function optimization. In 2012 IEEE Symposium on Computers and Communications (ISCC). 2012. IEEE.
2. Hu, X., Y. Shi, and R. Eberhart. Recent advances in particle swarm. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753). 2004. IEEE.
3. Eberhart, R. and J. Kennedy. A new optimizer using particle swarm theory. In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. 1995. Ieee.
4. Chen, S.-M., A. Sarosh, and Y.-F. Dong, Simulated annealing based artificial bee colony algorithm for global numerical optimization. Applied mathematics and computation, 2012. 219(8): p. 3575-3589.
5. Karaboga, D. and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 2007. 39(3): p. 459-471.
6. Dromey, R.G. and A.D. McGettrick, on specifying software quality. Software Quality Journal, 1992. 1(1): p. 45-74.
7. [7.] Sheta, A. Reliability growth modeling for software fault detection using particle swarm optimization. In 2006 IEEE International Conference on Evolutionary Computation. 2006. IEEE.
8. [8.] Fister, I., et al., A comprehensive review of firefly algorithms. Swarm and Evolutionary Computation, 2013. 13: p. 34-46.
9. [9.] Al-Qutaish, R.E., Quality models in software engineering literature: an analytical and comparative study. Journal of American Science, 2010. 6(3): p. 166-175.
10. [10.] Boehm, B. and V.R. Basili, Software defect reduction top 10 list. Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research, 2007. 34(1): p. 75.
11. [11]. AbouAssi, T.A., Using artificial bee colony to optimize software quality estimation models.(c2015). 2015.
12. [12.] Curtis, B., S.B. Sheppard, and P. Milliman. Third time charm: Stronger prediction of programmer performance by software complexity metrics. In Proceedings of the 4th international conference on Software engineering. 1979. IEEE Press.
13. [13.] Carnall, C.A., Managing change in organizations. 2007: Pearson Education.
14. [14.] Turhan, B. and A.B. Bener. Software Defect Prediction: Heuristics for Weighted Naïve Bayes. In ICSOFT (SE). 2007.
15. [15.] Li, P.L., et al., an Empirical Comparison of Field Defect Modeling Methods. 2005, Carnegie Mellon University2005.
16. [16.] Oliveira, B.R.d.N., A quality model for critical embedded systems. 2017, Universidade de São Paulo.
17. [17.] Xing, F., P. Guo, and M.R. Lyu. A novel method for early software quality prediction based on support vector machine. In 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05). 2005. IEEE.
18. [18.] Khoshgoftaar, T.M., et al., Predictive modeling techniques of software quality from software measures. IEEE Transactions on Software Engineering, 1992(11): p. 979-987.
19. [19.] Chidamber, S.R. and C.F. Kemerer, A metrics suite for object oriented design. IEEE Transactions on software engineering, 1994. 20(6): p. 476-493.
20. [20.] Evett, M., et al. GP-based software quality prediction. In Proceedings of the Third Annual Conference Genetic Programming, volume. 1998.
21. [21.] Afzal, W. and R. Torkar, on the application of genetic programming for software engineering predictive modeling: A systematic review. Expert Systems with Applications, 2011. 38(9): p. 11984-11997.
22. [22.] Wiper, M.P. and M.T. Rodríguez Bernal, Bayesian inference for a software reliability model using metrics information. 2001.

23. [23.] Wahono, R.S., A systematic literature review of software defect prediction. Journal of Software Engineering, 2015. 1(1): p. 1-16.
24. [24.] Sarro, F., A. Petrozziello, and M. Harman. Multi-objective software effort estimation. In 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). 2016. IEEE.
25. [25.] Yaremchuck, S., V. Kharchenko, and A. Gorbenko, Search of Similar Programs Using Code Metrics and Big Data-Based Assessment of Software Reliability, in Applications of Big Data Analytics. 2018, Springer. p. 185-211.
26. [26.] Singh, A., et al. A machine learning approach for modular workflow performance prediction. In Proceedings of the 12th Workshop on Workflows in Support of Large-Scale Science. 2017.
27. [27.] Capelli, F., et al., A genetic-algorithm-optimized fractal model to predict the constriction resistance from surface roughness measurements. IEEE Transactions on Instrumentation and Measurement, 2017. 66(9): p. 2437-2447.
28. [28.] Gharehchopogh, F.S. and A. Pourali, A new approach based on continuous genetic algorithm in software cost estimation. J. Sci. Res. Dev, 2015. 2(4): p. 87-94.
29. [29.] Bouktif, S., H. Sahraoui, and G. Antoniol. Simulated annealing for improving software quality prediction. In Proceedings of the 8th annual conference on Genetic and evolutionary computation. 2006.
30. [30.] Uysal, M. A Comparison of heuristic search algorithms for predicting the effort component of software projects. In 2008 International Conference on Computational Intelligence for Modelling Control & Automation. 2008. IEEE.
31. [31.] Vandecruys, O., et al., Mining software repositories for comprehensible software fault prediction models. Journal of Systems and software, 2008. 81(5): p. 823-839.
32. [32.] Azar, D. and J. Vybihal, An ant colony optimization algorithm to improve software quality prediction models: Case of class stability. Information and Software Technology, 2011. 53(4): p. 388-393.
33. [33.] Sehra, S.K., Y.S. Brar, and N. Kaur, Soft computing techniques for software effort estimation. arXiv preprint arXiv:1310.5221, 2013.
34. [34.] Kumari, S. and S. Pushkar, Performance analysis of the software cost estimation methods: a review. International Journal of Advanced Research in Computer Science and Software Engineering, 2013. 3(7).
35. [35.] Di Martino, S., et al. A genetic algorithm to configure support vector machines for predicting fault-prone components. In International conference on product focused software process improvement. 2011. Springer.
36. [36.] Sarro, F., et al. A further analysis on the use of genetic algorithm to configure support vector machines for inter-release fault prediction. In Proceedings of the 27th annual ACM symposium on applied computing. 2012.
37. [37.] Long, W., et al., An improved artificial bee colony with modified augmented Lagrangian for constrained optimization. Soft Computing, 2018. 22(14): p. 4789-4810.
38. [38.] Maleki, I., L. Ebrahimi, and M.K. Japelaghi, Ant Colony based Metaheuristic Algorithms for Software Cost Estimation. Computer Engineering, 2016. 1(1): p. 05-15.
39. [39.] Ghorbani, M., et al., Pan Evaporation prediction using a hybrid multilayer perceptron-firefly algorithm (MLP-FFA) model: case study in North Iran. Theoretical and applied climatology, 2018. 133(3-4): p. 1119-1131.
40. [40.] Mustaffa, Z., Y. Yusof, and S.S. Kamaruddin, Gasoline price forecasting: an application of LSSVM with improved ABC. Procedia-Social and Behavioral Sciences, 2014. 129: p. 601-609.
41. [41.] Briand, L.C. and J. Wüst, Empirical studies of quality models in object-oriented systems, in Advances in computers. 2002, Elsevier. p. 97-166.
42. [42.] Azar, D., H. Harmanani, and R. Korkmaz, A hybrid heuristic approach to optimize rule-based software quality estimation models. Information and Software Technology, 2009. 51(9): p. 1365-1376.