Abstract—
Software Reliability is turning into a basic possessions of any software system. It is a massive factor in software quality since it evaluates software drawbacks. Software defect prediction models have increased significant significance in accomplishing high software reliability. Software imperfection prediction display helps in early recognition of issues and add to their proficient evacuation and creating a dependable software system. In existing Software defect prediction system has been tested on fuzzy logic. Exact analyses have been completed and various methodologies have been created and proposed throughout the years, and many models have been proposed with various prescient abilities and proficiency. This paper introduces the review on existing data mining techniques utilized for prediction of software deserts. This paper will likewise present the idea of neural Networks which is been considered as one of the promising technique for prescient models.

Keywords— Software metrics, Method level metrics, Machine learning, Class level metrics, Defect prediction.

I. Introduction
The quality of software parts ought to be followed persistently amid the advancement of high-affirmation systems, for example, media transmission frameworks, therapeutic gadgets, and flight systems. Quality confirmation gathering can enhance the item quality by apportioning vital spending plan and HR to low quality modules related to various quality estimation models. Software defect, characterized as deviation from desire of software operation that may prompt software disappointments or any blemish identified with software itself, prompting colossal monetary misfortune, is a vital issue in software improvement life cycle [2]. As Software improvement is a human action, a considerable measure of deformities might be created amid the software advancement life cycle. It is very hard to create blame free, quality software due to expanding multifaceted nature and the limitations under which software is produced. Deficient Software postures impressive hazard by expanding the improvement and support expenses and client disappointment. Besides, software advancement organizations can't chance their business by giving flawed low quality software. It is, along these lines of extraordinary worry to find blame inclined software modules at a beginning time of the venture. Software defect prediction is the way toward finding faulty modules in software. For creating excellent software, the conveying last item ought to have as few imperfections as could reasonably be expected. For early location of software deformities could prompt lessened advancement costs and modify exertion and more solid software. In this way the defect prediction is vital to accomplish software quality. Imperfection prediction metrics assume the most essential part to fabricate a measurable prediction display. Most imperfection prediction metrics can be ordered into two sorts: code metrics and process metrics. The prediction models would then be able to be utilized by the software associations amid the early periods of software advancement to distinguish defect modules. The software associations can utilize this subset of metrics among the accessible extensive arrangement of software metrics. These metrics can be utilized as a part of building up the imperfection prediction models. Numerous scientists have utilized different strategies to build up the connection between the static code metrics and defect prediction. These techniques incorporate the conventional factual strategies, for example, calculated relapse and the machine learning techniques, for example, Decision trees, Naive Bayes, Support Vector Machines, Artificial Neural Networks This paper gives a basic audit of the different work completed in this field with the reason for recognizing future roads of research.
II. Literature Survey

The developing complexities of software and expanding interest of dependable software have prompted the advance of persistent research in the territories of viable software reliability evaluation. Software imperfection prediction is not another thing in software building area. Various Software Defect Prediction models and techniques have been proposed by various scientists as of late. In this segment, some imperative commitments around there are displayed.

A. Software Defect Prediction Based on Classification Techniques

Karunanithi et.al [1] exhibited the neural system display for software reliability prediction and found that neural system models are preferable at endpoint prediction over investigative models. They utilized distinctive systems like bolster forward NN, Jordan NN, intermittent neural systems.

Khoshgaftaar et.al [2] presented the utilization of the neural system as an instrument for anticipating software quality of a substantial media transmission system, characterizing modules into blame or non-blame inclined. They contrasted the Artificial Neural Network show and a non-parametric discriminant model, and found that Neural Network demonstrate has better prescient precision.

Kanmani et.al [3] presented two neural system based blame prediction models utilizing object arranged metrics and contrasted the outcomes and two measurable models utilizing five quality traits and inferred that neural systems improve.

B. Software Defect Prediction Based on Clustering Techniques

In the paper [4], creators proposed a novel software defect prediction technique in view of useful groups of projects to enhance the execution. Until at that point, most techniques proposed toward this path anticipate absconds by class or record. Tests completed presumed that bunch based models can altogether enhance the review from 31.6% to 99.2% and accuracy from 73.8% to 91.6%.

In the paper [5], k-implies based bunching approach has been utilized for finding the blame inclination of the Object arranged systems and found that k-implies based grouping techniques indicates 62.4% exactness. It likewise demonstrated high estimation of likelihood of location and low estimation of likelihood of false cautions. This investigation affirms the achievability and value of k implies based software blame prediction models.

C. Software Defect Prediction Based on Association control mining

In [6], analysts proposed prediction of imperfection affiliation and defect adjustment technique in light of affiliation control mining strategies. The proposed strategies were connected to abscond data comprising of more than 200 undertakings more than 15 years. It was finished up from trial comes about that exactness accomplished is high for both defect affiliation prediction and imperfection adjustment prediction. The outcomes got were likewise contrasted and PART, C4.5 and Naive Bayes technique and demonstrated the precision change by 23 percent.

In [7], scientists proposed a novel imperfection prediction demonstrate in light of social affiliation rules which are an augmentation of ordinal affiliation governs and depict numerical orderings between characteristics that normally happen over dataset. This proposed demonstrate was assessed on open source datasets and contrasted with comparable existing methodologies and found that this model over performed for the majority of the current machine learning based techniques for imperfection prediction.

D. Software Defect Prediction Based on Hybrid Approach

In the paper [8], a cross breed approach in light of K-Means Clustering and bolster forward neural system has been proposed and it was discovered that execution is better if there should be an occurrence of this half and half approach as contrasted and the current methodologies as far as exactness, mean total blunder and root mean square mistake esteems.

Mixture blame inclined module prediction technique was presented that consolidates affiliation control mining with calculated relapse investigation [9], [10]. On the off chance that a module fulfills the preface of one of the chose rules, the module is arranged by run as either blame inclined or not. Something else, the module is characterized by the strategic relapse. The prediction execution of this model was assessed and contrasted and three other blame inclined modules in
view of strategic relapse demonstrate, straight discriminant model and characterization tree. The trial comes about indicated change in execution when contrasted with regular strategies.[13, 14]

III. Issues and Challenges

Software prediction model only works well when enough amount of data is available in software repository within the organization to initially feed the model. Extraction of defects from software bug repository accurately is not done without a good data mining model. There is a need of good data mining model to predict the software defects from a bug repository.

a) Highly skewed and imbalanced datasets
- Existing prediction models based on un sampling as well as training dataset does not contain any information about number of fault per module and distribution of fault among modules.
- Data mining algorithms lack of business knowledge and hit a performance ceiling effect when cannot extract the additional information that related to software metrics with fault occurrence.
- Fit datasets are usually imbalanced that cause the degradation of defect prediction models.
- Highly skewed dataset is considered as the main cause of unsatisfactory prediction result. However the results of more balanced dataset are also unsatisfactory.

b) Early life cycle and multiple dataset
- Early life cycle data cannot be useful in identifying fault prone modules.
- No change in defect predictions results when different software repositories are mined.
- Single classifier is technically unfit to make use of all the features. However the problems of combing different classifier still remain unresolved.

c) Large number of features and high level software modules
- Most of the machine leading algorithms are not capable of extracting defects from the database that store continuous features.

- Supervised learning are useful for defect prediction at same logical levels but it is not suitable for high level software modules.
- Existing classifier based defect prediction model are insufficient accurate for practical use and use of a large number of features.

d) Accurate defect prediction model
- There is a need of accurate defect prediction model for large-scale software system which is more robust to noise.
- Traditional decision tree are used in classification of defective and non-defective modules. However traditional decision trees induction method contain several disadvantages.[10]
- There is a need of good data mining model to predict the software defects from a bug repository.
- Data transformation can improve the performance of software quality models.

e) Consistent data mining technique
- A good data mining technique to build a better prediction model is an open issue.
- Quality professional cannot find appropriate defect prediction techniques because there is no comparative study that asses the performance of these techniques.
- A good data mining technique to build a better prediction model is an open issue.
- Evaluation of different prediction model is still an open issue as well as effort reduction gain by using such model is ignored during evaluation.
- Various fault prediction techniques have been proposed but no one has proven to be consistently accurate. - A good data mining technique to build a better prediction model is an open issue.
- Different prediction techniques are used to assess the software quality but there is a lack of comparative study to evaluate the effectiveness of various models.

IV. Problem Definition

As we have discussed upon earlier, defect prediction is vital in nature. Our prime objective is to predict defects without overrunning the estimated cost as well
as without delaying scheduled delivery of software. However, the main issue related to this is mainly the plethora of models which can be used for the same. All models of defect prediction have their own set of advantages and disadvantages which makes it hard to understand which fault prediction model should be used and more importantly in what type of project. Since every project tends to be unique, this is hard from a decision making standpoint. However, we believe thorough model evaluation can enable project managers to make a more informed decision. [19]

In our study, we will cover the popular models of defect prediction and evaluate the pros and cons of each model along with the situations where the models can be used. We will evaluate the models based a varied set of criteria depending on the model being discussed. After evaluation, we will also include our personal observations and interpretations on why we think certain decision models are useful along with substantiating case studies of real world usage wherever possible. [21]

V. Software Defect Prediction Models

Prediction Model using size and complexity metrics

Among the popular models of defect prediction, the approach that uses size and complexity metrics is fairly well known. This model uses the program code as a basis for prediction of defects. More specifically, lines of code (LOC) are used along with the concept of complexity model developed by McCabe. Using regression equations, simple prediction metrics estimates can be obtained using a dependent variable (D) defined as the sum of defects found during testing and after 2 months post release. Famously, Akiyama made 4 equations. We have illustrated the equation that includes the LOC metric:

\[
\text{Defect} (D) = 4.86 + 0.018 \text{Lines of Code (L)} \tag{1}
\]

Gaffney deduced above equation (1) into another prediction equation. He argued that LOC was not language dependent owing to optimal size for individual modules with regards to defect density. The regression equation is given below:

\[
D = 4.2 + 0.0015 L_{V3} \tag{2}
\]

The size and complexity models presume that defects are direct function of size or defects are occurred due to program complexity. This model ignores the underlying casual effects of programmers and designers. They are the human factors who actually commence the defects, so any attribution for flawed code depends on individual(s) to certain extent. Poor design capability or problem difficulty may result in highly complex programs. Difficult problems might require complex solutions and naive programmers might create „spaghetti code” [22].

Machine Learning Based Models

Machine learning (ML) algorithms has demonstrated great practical significance in resolving a wide range of engineering problems encompassing the prediction of failure, error, and defect-impulsions as the system software grows to be more complex. ML algorithms are very useful where problem domains are not well defined, human knowledge is limited and dynamic adaption for changing condition is needed, in order to develop efficient algorithms. Machine learning encompasses different types of learning such as artificial neural networks (ANN), concept learning (CL), Bayesian belief networks (BBN), reinforcement learning (RL), genetic algorithms (GA) and genetic programming (GP), instance-based learning (IBL), decision trees (DT), inductive logic programming (ILP), and analytical learning (AL)[23].

G. John, P. Langley [24] employed RF method for prediction of faulty modules with NASA data sets. Prediction of software quality was introduced by Khoshgoftaar et al. [25] by using artificial neural network. In this model they classified modules as fault prone or non-fault prone, using large telecommunication software system. They compared their end results with another non–parametric model achieved from discriminant method. Fenton et al. [26] suggested the use of Bayesian belief networks (BBN) for the prediction of faulty software modules. Elish et al. [27] recommended the use of support vector machines for predicting defected modules with context of NASA data sets. This model compares its prediction performance with other statistical and machine learning models. We have discussed few models in detail to enhance the understanding of Machine learning based prediction models.

The Probabilistic Model for Defect Prediction using Bayesian Belief Network

Fenton, Krause and Neil [6] proposed a probabilistic model for defect prediction. They recommended a
A holistic model rather than a single issue (for e.g. size, or complexity, or testing metrics, or process quality data) model, by combining the different factors of casual evidence in order to successful defect prediction. The model uses Bayesian Belief Network (BBN) as the suitable practice for representation of this evidence. The Bayesian approach causes statistical conclusion to be improved by expert judgment in those parts of a problem sphere where empirical data is scattered. Additionally, the causal or influence organization of the model better reflects the series of real world events and relations than any other practice. [15, 16]

BBN can be exploited to support effective decision making for SPI (Software Process Improvement), by executing the following steps.

Identification of variables (Hypothesis, Information or Mediating variables)

Defining the accurate relationships among variables

Achieve a probability distribution for each variable in the BBN

Fig. 1 Bayesian Approach

A BBN represents the joint probability distribution for a set of variables. This is achieved by defining Directed acyclic graph (DAG) and Conditional probability tables A BBN can be employed to deduce the probability distribution for a target variable (e.g., “Defects Detected”), which indicates the probability that the variable will obtain on each of its possible values (e.g., “very low”, “low”, “average”, “high”, or “very high” for the variable “Defects Detected”) given the observed values of the other variables [8,9].

N. Fenton, M. Neil and D. Marquez [7] reviewed the use of Bayesian networks to overcome impediments of using BN"s for predicting software defects and software quality. BN tools and algorithms suffered from „Achilles” heel. This compelled modelers to redefine discretization intervals in advance and resulted in inadequate predictions for large set of data. To improve this „dynamic discretization” algorithm was used. This algorithm exploits entropy error as the basis for approximation allowing more accuracy.

The Probabilistic Model for Defect Prediction using Fuzzy Logic Model

The Fuzzy Logic model is based on the concept or reasoning and works on a value that is approximate in nature. It is a step up from conventional Boolean Logic where there can only be True or False. In case of Fuzzy logic, the truth of any statement is degree and not an absolute number. Modeled on human intuition and behavior, the biggest plus point of Fuzzy logic is that as opposed to the traditional yes – no answers, this model factors in the degree of truth and hence makes allocation for the more human like answers. [11, 12]

Previously in this report, we have elaborated on why it is important to identify software quality issues at an early stage. Ajeet Kumar Pandey and N. K. Goyal [10] suggested the model of Fuzzy Logic and the software metrics as well as process maturity, the model can be constructed as follows:

Fig. 2 Fuzzy Logic Approach

This model uses inputs and puts them in a range system. After this, a set of rules is defined that dictates and influences how inputs will be utilized in getting the output as well as finding the definitive value in the fuzzy set. The model has a set of metrics or reliability relevant metric (RRML) list which is made from the available software metrics. The metrics are pertinent
to their respective phases in the software development life cycle.

Requirement Phase Metrics - As you can see the model has uses three requirements metrics (RM) i.e. Requirements Change Request (RCR), Review, Inspection and Walk through (RIW), and Process Maturity (PM) as input to the requirements phase.

Design Phase Metrics – similar to the above phase, three design metrics (DM) i.e. design defect density (DDD), fault days number (FDN), and data flow complexity (DC) have considered as input.

Coding Phase Metrics – In this phase, two coding metrics (CM) such as code defect density (CDD) and cyclomatic complexity (CC) have been taken as input at coding phase. The outputs of the model will be the number of faults at the end of Requirements Phase (FRP), number of Faults at the end of Design Phase (FDP), and number of Faults at the end of Coding Phase (FCP). [17, 18]

Defect Prediction Models Based on Genetic Algorithms

Genetic Algorithms is an approach to machine learning which behaves similarly to the human gene and the Darwinian theory of natural selection. It is a part of the Evolutionary Algorithms which generate solutions based on the techniques more commonly found in nature like mutation, selection, crossover etc.

Genetic Algorithms are implemented beginning with an individual population that is usually represented in the form of trees. A possible solution is represented by each tree or say chromosome in this case. Nodes on the tree signify particular traits that relates to the problem for which the solution is being searched. Collectively, the set of potential solutions to the problem is (represented by the chromosomes) as known as the population. [20]

Where genetic algorithms come into place is when you need to solve problems which can have many solutions. Here, genetic algorithms are being used to cluster the classes defined as per object oriented metrics into subsystems or commonly known as components of software. As elaborated earlier, genetic algorithm uses an approach akin to Charles Darwin’s “Survival of the Fittest” or natural selection. The reason this approach is being considered is because the large solutions set which provide a number of possible solutions to a problem. When applying a genetic algorithm to a problem, there are a few implications which are made. The same are as follows

a) There must be a fitness function present for the evaluation of weather a solution is a possible one or not

b) Whenever there is a solution found, there should a representation of it made by a chromosome.

c) Whichever genetic operators will be applied must be established

Additionally the definition of a solution in this case would be one which would be both complete as well as valid. In terms of a representation, there is the assumption that the possible solutions have been encoded in the solutions space.

How do Genetic Algorithms work?

In the beginning, the Genetic Algorithms start with a large population. In that population, each individual represents a plausible solution to the problem. These individuals in the population are then encoded in a binary string that is called a chromosome. After that, the group of the individuals will compete so that they can reproduce and then formulate the next generation. However, there is a function called the fitness function that determines which of the competing individuals will gain the right to reproduce. Having the fitness function in place makes sure that only the best individuals of the population will be able to carry over their offspring into the next generation. The next generation is formed by the following activities taking place.

a) Reproduction – reproduction process takes place when two chromosomes exchange a portion of their code to form the new individuals. The crossover points (where the bits of the code will exchange) are selected by random (for a simple version of the algorithm). At the crossover point, the chromosomes exchange the data keeping the original data up to that point.

b) Mutations – this comes in to introduce variation in the next generation which prevents the reaching of local minima. Whereas crossover alters the genes after a randomly selected crossover point between 2 chromosomes, mutation selects on node in the tree of one chromosome and changes the genetic material.
This process repeats itself until there is a perfect solution set reached (optimal fitness level). However, there are occasions when this does not happen. In such cases, the program terminates after a set of iterations. The iterations of the proceeds are also known as generations. [17]

Example of using Genetic Algorithms in a Web Fault Prediction

Research on Genetic Algorithms being applied is few since this is a relatively new domain. In the following, we show how it can be applied to an online web application, proposed by Marshima M. Rosli et al. [6].

Namely, there is the requirement of three components to build the model of Fault prediction using Genetic Algorithms. They are as follows:

a) Software Metric Extractor
b) Fault Classes Detection System
c) Genetic Algorithm Generator

The model can be represented diagrammatically as follows:

![Diagram of the Genetic Algorithm Approach](image)

Fig 3 Genetic Algorithm Approach

How this model would work is that the information about metrics would be extracted from source files as well as the logs. Then the optimal metrics can be found by the GAG part of the model which will subsequently use reproductions and mutations to create new generation of populations until the optimal set of metrics are found.

Software Defect Prediction Models using Artificial Neural Network

The artificial neural network is based on the human biological system in its architecture and design. It processes information in a way similar to the human brain using an intricate system of interconnected neurons to solve highly complex problems. The artificial neural networks work in a similar fashion and use a trial and error process to construct models of the problem space. Using “guesses” of what the desired output should be, the actual result and the predicted result (guesses) are compared and if there is a difference, that value is passed on to the network as feedback so that internal adjustments can be made to get a better quality of results in the future. Over a period of time, this process continues as the network is presented with other sets of data until it gives an accurate model of the process.

As told in the introduction, the artificial neural network has a set of elements which perform the computations required on the problem set. In this case, the feed forward as well as the back propagation training algorithm have been used for the purposes of defect prediction. The architecture of the network is such that there are two neurons in the output layer (basically fault and non-fault). The output that has the greatest value is selected thereafter. The learning process happens by finding a vector of the connection weights which lower the error sum squared on the training set. The training of the network happens with the continuous back propagation and the weights are adjusted after each observation which is then fed forward for each of the classes (fault and non-fault).

Neha Gautam, Parvinder S. Sandhu, Sunil Khullar [1] recommended to use Multilayer Perceptron and RBF based Neural Network approaches for the identification of the relation between the several qualitative as well as quantitative factor of the modules. These approaches also identify the number of defects existing in the module that will be beneficial for the prediction of defects. The methodology consists of the following steps:

1. Find the Qualitative and Quantitative attributes of software systems
2. Select the suitable metric values as representation of statement

©www.ijrcct.org | * Corresponding Author
3. Analyze, refine metrics and normalize the metric values and Explore different Neural Network Techniques

VI. Conclusion

Defects can assess in directing the software quality assurance measures as well as improve software management process if developers find and fix them early in the software life cycle. Effective Defect prediction is based on good data mining model. In this we surveyed different data mining algorithms used for defect prediction. We also discuss the performance and effectiveness of data mining algorithms. This survey also has showed that all the issues for selecting a data mining technique for defect prediction and their provided solutions have been discussed. Our most important finding is that there is no single data mining technique that is more powerful or suitable for all type of projects. In order to select a better data mining algorithm, domain expert must consider the various factors like problem domain, type of data sets, nature of project, uncertainty in data set etc. Multiple classifiers were combined by majority voting of experts to get more accurate result. Our findings indicate that early life cycle data can be highly effective in defect prediction. However, a combination of different data sources can utilize to get better prediction results. Another finding of this paper is Sampling method are useful to improve performance when dataset are highly skewed. Data transformation cannot improve the performance of defect prediction. Integration of discretization method with classification algorithm improves the defect prediction accuracy by transforming the continuous features into discrete features. However different techniques are applied in identifying defects for large features and high level software modules.

VI. Future Work and Open Issues

Future work in this area should:

- Study other unsupervised or semi supervised learning framework and compare the performance of other different data mining algorithms to find out the best algorithm for defect prediction.

- Analyze the effect of classifier with feature selection and find out whether the cost sensitive learning algorithms can be used to build better defect prediction models.

References


Authors

**T. Ravi Kumar** received M.Tech degree in Computer Science and Engineering from Jawaharlal Nehru technological University Hyderabad, A.P., India. And B.tech in computer Science and Information Technology from Jawaharlal Nehru Technological University Hyderabad, A.P., India. He is having 10 years of experience in teaching and presently working as Sr Assistant Professor in the Department of Computer Science and Engineering at Aditya Institute of Technology and Management, Tekkali [AITAM], A.P., India. His area of research includes Software Engineering, Fuzzy logic and Software Testing Methodologies.

**Dr T. Srinivas Rao** received B.Tech degree from GITAM, Andhra University, Visakhapatnam A.P., India. Received M.Tech degree from Andhra University, Visakhapatnam, A.P., India. Received Ph.D. degree from Andhra University, Visakhapatnam, A.P., India. Presently he is working as Associate professor, department of CSE, Gitam Institute of Technology, GITAM University, Visakhapatnam. His research interest includes wireless communication (WiFi, WiMax), Mobile Ad hoc Networks, Sensor Networks, Neural Networks and fuzzy logic, Communication networks, Data mining, software engineering, Machine Learning.