IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 1, January 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201028 |

Software Defect Prediction using Machine Learning Techniques: A Systematic Review

Seema Rani[1], Prof. Suresh. S. Gawande[2]

M. Tech. Scholar, Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal, India[1]

Guide, Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal, India[2]

ABSTRACT- Software testing is the process of finding faults in software while executing it. The results of the testing are used to find and correct faults. Software defect prediction estimates is where faults are likely to occur in source code. The results from the defect prediction can be used to optimize testing and ultimately improve software quality. Machine learning, that concerns computer programs learning from data, is used to build prediction models which then can be used to classify data. Many researchers have already been working in the field of defect prediction in software using some machine learning algorithms. Their results vary from dataset to dataset. These algorithms give inconsistent output for predicting defects in a random software project. Researchers have not decided which machine learning algorithm is best suitable for correctly predicting the defects in software so recent developments in machine learning introduce ensembling methods to predict defects. Ensembling takes the advantages of different techniques to give a better prediction of defects compared to individual base models. In this paper the study of different machine learning algorithm for software defect prediction.

KEYWORDS- Software Testing, Machine Learning, Ensembling Technique

I. INTRODUCTION

With growing demand and technology, the software industry is rapidly evolving. Since humans do most of the software development, defects will inevitably occur. In general, defects can be defined as undesired or unacceptable deviations in software documents, programs, and data [1].Defects may exist in requirements analysis because the product manager misinterprets the customer's needs, and as a result, this defect will also carry on to the system design phase. Defects may also occur in the code due to inexperienced coders. Defects significantly impact software quality, such as increased software maintenance costs, especially in healthcare, and aerospace software defects can have serious consequences. If the fault is detected after deployment, it causes an overhead on the development team as they need to re-design some software modules, which increases the development costs. Defects are nightmares for reputed organizations. Their reputation is affected due to customer dissatisfaction and hence reduces its market share. Therefore, software testing has become one of the main focuses of industrial research [2]. With the rise in software development and software complexity, the number of defects has increased to the extent that traditional manual methods consume much time and become inefficient. The rise of machine learning has made automatic classification of defects a research hotspot. In this paper, we initially discuss software defects in detail and their different categories available in the literature and then discuss the manual classification methods proposed by various researchers. Finally, we present the analysis of the state of the art machine learning algorithms for automatic software detection [3, 4].

II. SOFTWARE DEFECT PREDICTION

Software defect prediction (SDP) is a technique for improving software quality and reducing software testing costs through the creation of multiple categorization or classification models utilizing various machine learning approaches. Many companies that develop various types of software want to foresee problems to maintain software quality for customer satisfaction and save testing costs. SDP is part of the software development life cycle in which we predict the fault using a Machine Learning (ML) approach with historical data [5]. It is a structured methodology that enables the creation of high-quality, low-cost software in the short possible time to meet customer expectations.

SDP's mission is to provide high-quality software and dependability while making efficient use of limited resources. As a result of this, software developers will be able to prioritize the utilization of computer resources at each level of the software development process [6, 7]. Many organizations which are producing various types of software wish to predict the defects in software to maintain software quality for customer satisfaction and to save the testing cost. SDP is

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201028 |

used to increase the software's quality and testing can be conducted efficiently by constructing various classification models using different machine learning methods. A wide range of ML approaches has been investigated so far to anticipate errors in software modules to enhance software quality and reduce software testing costs. There are several ML techniques, which are implemented in the SDP Decision tree, Naïve Bayes (NB), Radial basis function, Support Vector Machine (SVM), K-nearest neighbor, Multi-layer perceptron and Random Forest (RF) [4].

New advances in ML are ensembling techniques and various ML techniques with feature selection methods such as PCA, etc. In the extant literature, there are several types of software metrics that have been found and utilized for SDP. It would be more practical to deal with the most important software metrics and focus on them to predict defect in software [7]. SDP analyses data from the past acquired from software repositories to find out the quality and reliability of software modules [8]. There are numerous types of software metrics that have been found and utilized for SDP in existing literature. SDP models are generated with the help of software metrics from data acquired from previously established systems or similar software initiatives [9].

It would be more practical to look at and focus on the most important software metrics to predict bugs in the software. Therefore, the dataset used in the paper has been publicly available on the Promise Repository since 2005, providing information on various applications that NASA (National Aeronautics and Space Administration) has investigated. In the research context, after dataset pre-processing and feature selection (FS), K-means clustering is used to perform the output categorization. Then, ML approaches such as SVM, NB and RF with and without particle swarm optimization (PSO) are used. An ensemble approach is then used to integrate the results. Finally, all ML models are analysed and compared to the previous studies. The models' performance is evaluated using precision, accuracy, recall, F-measure, performance error metrics, and confusion matrix.

Software deformity (or deficiency) prediction is viewed as one of the most practical and furthermore useful device which let us know whether a specific module is having imperfection or not. Software professionals consider it to be an essential stage for guaranteeing the nature of the procedure or the item which is to be created. It made light of an exceptionally pivotal job in achieving the cases the software industry that it can't meet the necessities in the spending plan and on schedule [10]. The colossal venture and cash spent on software designing improvement prompts an increment in software framework support costs. Today, the huge size of the software created is getting progressively mind boggling. Countless program codes, as well. To this end, the likelihood of software insufficiencies has been expanded and strategies for quality affirmation are not adequate to defeat all software lacks in colossal frameworks. One of the viable method to improve the nature of software is to anticipate software abandons, which is additionally a powerful method to alleviate the exertion of assessing or testing software code [13]. Under this situation, just piece of the software antiquities should be reviewed or tried and the remaining ones disregarded. Settling an imperfection, or flaw, prompts exponential increments in the event that it enters to the resulting stages of a software advancement lifecycle. The benefit of distinguishing software deficiencies in the underlying stages not just yields less blames and an upgraded software w.r.t quality, yet in addition helps in creating of a practical model. Likewise, we just spotlight on the more vulnerable modules during testing and upkeep stages which in the long run prompts powerful advancement of model. Subsequently, the constrained assets of an association could be sensibly apportioned with the target of identification and rectification of the most extreme number of software abandons. In this manner, this subject of prediction of the software shortcomings has been dissected widely and a ton of strategies have been recommended to address this issue [14].

III. LITERATURE REVIEW

Iqra Mehmood et al. [1], defect prediction is one of the most active areas within the software engineering community. The software's success depends on closing the gap between software engineering and data mining. Programming abandons expectation figures the source code blunders before the testing stage. The effect area in software is frequently explored using methods for predicting software defects like clustering, statistical methods, mixed algorithms, metrics based on neural networks, black box testing, white box testing, and machine learning. The primary commitment of this exploration is the utilization of element choice interestingly to expand the exactness of AI classifiers in deserts pre-lingual authority. The goal of this study is to further develop the deformities expectation exactness in five informational indexes of NASA to be specific; CM1, JM1, KC2, KC1, and PC1. These NASA informational collections are available to public. In this exploration, the element determination procedure is use with AI methods; To achieve higher defect prediction accuracy than without feature selection (WOFS), we used Random Forest, Logistic Regression, Multilayer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Neural Networks, and Decision Stump. The machine-learning tool WEKA (Waikato Environment for Knowledge Analysis) is used in the research workbench to refine da-ta,

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201028 |

preprocess data, and apply the classifiers mentioned earlier. A mini tab statistical tool is used to evaluate statistical analyses. In contrast to the accuracy of WOFS, this study reveals that defects prediction with feature selection (WFS) is more accurate.

L.-Q. Chen et al. [2], SDP has emerged as an essential component of software testing and ensures the delivery of high-quality software products. Programming deformity forecast is separated into conventional programming imperfection expectation and without a moment to spare programming imperfection forecast (JIT-SDP). In any case, the vast majority of the current programming imperfection expectation systems are somewhat improved, which makes it very hard to furnish designers with more definite reference data. This paper proposes a software defect prediction framework based on heterogeneous feature selection and Nested-Stacking to improve software defect prediction and allocate testing resources efficiently. There are three stages to the framework: evaluation of model classification performance, Nested-Stacking classifier, and data set preprocessing and feature selection. The framework's novel heterogeneous feature selection and nested custom classifiers have the potential to significantly raise software defect prediction accuracy. The AUC and F1-score, two comprehensive evaluation indicators, are used in this paper to demonstrate the classification performance of the model on two software defect data sets (Kamei and PROMISE). The investigation did enormous scope inside project imperfection expectation (WPDP) and cross-project deformity forecast (CPDP). According to the findings, the proposed framework outperforms the baseline models in terms of classification performance when applied to the two types of software defect data sets.

M. Pavana et al. [3], programming shortcoming forecast (SFP) has an essential impact in nature of programming. It aids in the early stages of the software development life cycle (SDLC) in identifying flawed constructs. When the predicted results do not match the actual output, this software defect is also known as a defect. This can be alluding as a blunder, shortcoming, bug in a PC program. The topic of machine learning (ML) deals with the process of creating or designing computer programs that constantly improve their effectiveness at specific tasks through experience. In field of computer programming, AI assumes a vital part and contains various methodologies like test exertion expectation and cost forecast. Software faults prediction (SFP) is the most widely studied of these prediction techniques in software engineering. Machine learning techniques like Naive Bayes (NB), support vector machine (SVM), logistic regression (LR), and random forest (RF) were used to make the predictions in this paper. When selecting highly correlated input variables, a feature selection method like Spearman's rank correlation is used. Dimensionality decrease techniques, for example, LDA and PCA are utilized for diminishing the aspects. These algorithms are evaluated and performed with accuracy, precision, and recall.

A. Al-Nusirat et al. [4], by focusing on defect modules, software defect prediction is a practical way to increase the quality, efficiency, and cost of software testing. The dataset of programming imperfection expectation normally has a class unevenness issue with not many deficient modules contrasted with non-faulty modules. The Neural Network suffers as a result of this circumstance, which can result in inaccurate fitting and overfitting. Engineered Minority Over-inspecting Procedure (Destroyed) is one of the famous strategies that can take care of the issue of class irregularity. However, the user must determine the hyperparameters for both SMOTE and Neural Networks prior to modeling. In this review, we applied the Brain Organizations Based Destroyed, a blend of Brain Organization and Destroyed with each hyperparameter of Destroyed and Brain Organization that are upgraded utilizing irregular pursuit to take care of the class lopsidedness issue in the six NASA datasets. The outcomes utilize a 5*5 cross-approval show that expands Bal by 25.48% and Review by 45.99% contrasted with the first Brain Organization. SMOTE based on neural networks and SMOTE based on "traditional" machine learning are also compared in terms of performance. In terms of average rank, the neural network-based SMOTE comes out on top.

Mumtaz, B et al. [5], Software Defect Prediction (SDP) is a dynamic research field in the software industry. A quality software product results in customer satisfaction. However, the higher the number of user requirements, the more complex will be the software, with a correspondingly higher probability of failure. SDP is a challenging task requiring smart algorithms that can estimate the quality of a software component before it is handed over to the end-user. In this paper, we propose a hybrid approach to address this particular issue. Our approach combines the feature selection capability of the Optimized Artificial Immune Networks (Opt-aiNet) algorithm with benchmark machine-learning classifiers for the better detection of bugs in software modules. Our proposed methodology was tested and validated using 5 open-source National Aeronautics and Space Administration (NASA) data sets from the PROMISE repository: CM1, KC2, JM1, KC1 and PC1. Results were reported in terms of accuracy level and of an AUC with highest accuracy, namely, 94.82%. The results of our experiments indicate that the detection capability of benchmark classifiers can be improved by incorporating Opt-aiNet as a feature selection (FS) method.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201028 |

R. Bahaweres et al. [6], various programming imperfection expectation models have been proposed to work on the nature of programming throughout recent many years. An inexorably famous methodology is to utilize AI. There are two types of these strategies: supervised methods, in which the training data must be labeled, whether they are faulty or not, and unsupervised methods, in which the data do not require labeling. Regulated forecast models prevail. Notwithstanding, by and by it is frequently hard to gather deformity order names to prepare a Managed Imperfection Forecast (SDP) model. Consequently, Unsupervised Defect Prediction (UnSDP) models have begun to gain attention in recent years.

N. Li. et al. [7], the principal point of our orderly survey is to furnish programming experts and specialists with direction for programming imperfection expectation, especially in regards to whether the utilization of solo expectation models is a feasible choice. We look at 49 primary UnSDP studies that meet our inclusion criteria. We look into which unsupervised learning algorithms were used and how well supervised and unsupervised models predicted each other from these primary studies.

Cetiner, M. et al. [8], in the Software Engineering concept, the prediction of the software defects plays a vital role in increasing the quality of the software systems, which is one of the most critical and expensive phases of the software development lifecycle. While the use of software systems is increasing in our daily lives, their dependencies and complexities are also increasing, and this results in a suitable environment for defects. Due to the existence of software defects, the software produces incorrect results and behaviors. What is more critical than defects, is finding them before they occur. Therefore detection (and also prediction) of the software defects enables the managers of the software to make an efficient allocation of the resources for the maintenance and testing phases. In the literature, there are different proposals for the prediction of software defects. In this paper, we made a comparative analysis about the machine learning-based software defect prediction systems by comparing 10 learning algorithms like Decision Tree, Naive Bayes, K-Nearest Neighbor, Support Vector Machine, Random Forest, Extra Trees, Adaboost, Gradient Boosting, Bagging, and Multi-Layer Perceptron, on the public datasets CM1, KC1, KC2, JM1, and PC1 from the PROMISE warehouse. The experimental results showed that proposed models result in proper accuracy levels for software defect prediction to increase the quality of the software.

Y. Qiu et al. [9], numerous programming assignments vigorously depend available created programming highlights, for example , imperfection expectation, weakness revelation, programming prerequisites, code audit, and malware recognition. Past answers for these undertakings typically straightforwardly utilize the hand-created highlights or component choice methods for grouping or relapse, which as a rule prompts sub-standard outcomes because of their absence of strong portrayals of the hand-made highlights. This paper uses the effort-aware just-in-time software defect prediction (JIT-SDP), a typical hand-crafted feature-based task, as an example to explore new potential solutions to the aforementioned issue. Neural forest (NF) is a new model we propose that builds on deep neural networks and decision forests to create a comprehensive system for automatically exploring powerful feature representations used in the following classification. In the beginning, NF makes use of a deep neural network to learn brand-new feature representations from manually crafted features. Following the neural network, a decision forest is connected to perform classification and direct feature representation learning simultaneously. NF predominantly targets taking care of the difficult issue of consolidating the two unique universes of brain organizations and choice timberlands in a start to finish way. For within- and cross-project defect prediction, NF achieves significantly better results than previous state-of-the-art defect predictors and five designed baselines on six well-known benchmarks. The classification problems that rely on the hand-crafted features can be applied to the proposed NF model.

C. Manjula et al. [10], numerous methods, including data mining and machine learning, have been developed for the early prediction of software defects. Still early forecast of deformities is a difficult errand which should be tended to and can be improved by getting higher order pace of imperfection expectation. We present a hybrid strategy that combines a genetic algorithm (GA) for feature optimization with a deep neural network (DNN) for classification in order to address this issue. A better variant of GA is integrated which incorporates another strategy for chromosome planning and wellness capability calculation. DNN method is additionally ad libbed utilizing versatile auto-encoder which gives better portrayal of chosen programming highlights. Through case studies, the proposed hybrid approach's increased efficiency as a result of using an optimization technique is demonstrated. Using the MATLAB tool, an experimental study on software defect prediction is conducted taking into account the PROMISE dataset. In this review, we have involved the proposed novel technique for characterization and imperfection expectation. According to a comparison study, the proposed method for predicting software defects outperforms other methods with an accuracy of 97.82 percent for the KC1 dataset, 97.59 percent for the CM1 dataset, 97.96 percent for the PC3 dataset, and 98.00 percent for the PC4 dataset.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201028 |

IV. MACHINE LEARNING

Machine Learning is a subset of Artificial Intelligence concerned with “teaching” computers how to act without being explicitly programmed for every possible scenario. The central concept in Machine Learning is developing algorithms that can self-learn by training on a massive number of inputs. Machine learning algorithms are used in various applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks [4]. Machine learning enables the analysis of vast amounts of information. While it usually delivers faster, more precise results to identify profitable prospects or dangerous risks, it may also require additional time and assets to train it appropriately. Merging machine learning with AI and perceptive technologies can make it even more effective in processing vast volumes of information. Machine learning is closely associated with computational statistics, which focuses on making predictions using computers. Machine learning approaches are conventionally divided into three broad categories, namely Supervised Learning, Unsupervised Learning & Semi-supervised Learning, depending on the nature of the "signal" or "feedback" available to the learning system.

Face anti-spoofing (FAS) has lately attracted increasing attention due to its vital role in securing face recognition systems from presentation attacks (PAs). As more and more realistic PAs with novel types spring up, traditional FAS methods based on handcrafted features become unreliable due to their limited representation capacity. With the emergence of large-scale academic datasets in the recent decade, machine learning based FAS achieve remarkable performance and dominate this area.

Supervised Learning

A model is trained through a process of learning in which predictions must be made and corrected if those predictions are wrong. The training process continues until a desired degree of accuracy is reached on the training data. Input data is called training data and has a known spam / not-spam label or result at one time.

Unsupervised Learning

By deducting the structures present in the input data, a model is prepared. This may be for general rules to be extracted. It may be through a mathematical process that redundancy can be systematically reduced, or similar data can be organized. There is no labeling of input data, and there is no known result.

Semi-Supervised Learning

Semi-supervised learning fell between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). There is a desired problem of prediction, but the model needs to learn the structures and make predictions to organize the data. Input data is a combination of instances that are marked and unlabeled.

V. CONCLUSION

Early detection and prediction of software defects have a significant effect on software quality measurement. A software quality prediction model relies on the information mined from software measurement data and thus the better the selection of software metrics is by eliminating the redundant and less important features before training any defect prediction model the more probable it is to provide a more accurate end result. In this particular study, we examined the consequences of the FS technique on the overall performance accuracy of SDP. The main motive of this study was to examine the various classification techniques and ensemble techniques in order to find which method gives better fault prediction performance. Thus we employed four classification techniques (naive bayes, KNN, neural network and support vector machine). Out of these 4, we selected the best 3 techniques on the basis of AUC values.

REFERENCES

[1] Iqra Mehmood, Sidra Shahid, Hameed Hussain, Inayat Khan, Shafiq Ahmad, Shahid Rahman, Najeeb Ullah and Shamsul Huda, “A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning”, IEEE Access 2023.

[2] L.-Q. Chen, C. Wang, and S.-L. Song, ‘‘Software defect prediction based on nested-stacking and heterogeneous feature selection,’’ *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3333–3348, Aug. 2022

[3] M. Pavana, L. Pushpa, and A. Parkavi, ‘‘Software fault prediction using machine learning algorithms,’’ in *Proc. Int. Conf. Adv. Elect. Comput. Technol.*, 2022, pp. 185–197

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201028 |

[4] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. Al-Ayyoub, and N. Al-Dhfairi, ''Dynamic detection of software defects using supervised learning techniques,'' *Int. J. Commun. Netw. Inf. Secur.*, vol. 11, no. 1, pp. 185–191, Apr. 2022.

[5] Mumtaz, B.; Kanwal, S.; Alamri, S.; Khan, F. Feature selection using artificial immune network: An approach for software defect prediction. Intell. Autom. Soft Comput. 2021, 29, 669–684.

[6] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, ''Software defect prediction using neural network basedSMOTE,'' in *Proc. 7th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, Oct. 2020, pp. 71–76

[7] N. Li, M. Shepperd, and Y. Guo, ''A systematic review of unsupervised learning techniques for software defect prediction,'' *Inf. Softw. Technol.*, vol. 122, Jun. 2020, Art. no. 106287.

[8] Cetiner, M.; Sahingoz, O.K. A Comparative Analysis for Machine Learning based Software Defect Prediction Systems. In Proceedings of the 2020 11th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1–3 July 2020; pp. 1–7.

[9] Y. Qiu, Y. Liu, A. Liu, J. Zhu, and J. Xu, ''Automatic feature exploration and an application in defect prediction,'' *IEEE Access*, vol. 7, pp. 112097–112112, 2019.

[10] Manjula, C.; Florence, L. A Deep neural network based hybrid approach for software defect prediction using software metrics. Clust. Comput. 2019, 22, 9847–9863.

[11] A. Alsaeedi and M. Z. Khan, ''Software defect prediction using supervised machine learning and ensemble techniques: A comparative study,'' *J. Softw. Eng. Appl.*, vol. 12, no. 5, pp. 85–100, 2019.

[12] C. Manjula and L. Florence, ''Deep neural network based hybrid approach for software defect prediction using software metrics,'' *Cluster Comput.*, vol. 22, no. S4, pp. 9847–9863, Jul. 2019.

[13] R. Jayanthi and L. Florence, ''Software defect prediction techniques using metrics based on neural network classifier,'' *Cluster Comput.*, vol. 22, no. S1, pp. 77–88, Jan. 2019.

[14] Iqbal, A.; Aftab, S.; Ali, U.; Nawaz, Z.; Sana, L.; Ahmad, M.; Husen, A. Performance analysis of machine learning techniques on software defect prediction using NASA datasets. Int. J. Adv. Comput. Sci. Appl. 2019, 10, 300–308.

[15] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, ''Software bug prediction using machine learning approach,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details