International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Deep Learning for Software Defect Prediction: An LSTM-based Approach

Prashant Sahatiya
Department of Computer Applications
Centre for Distance and Online Education
Parul University
Vadodara, India
prashant.sahatiya30784@paruluniversity.ac.in

Dr. Harshal Shah
Department of Computer Science & Engineering
Parul Institute of Technology
Parul University
Vadodara, India
harshal.shah@paruluniversity.ac.in

Abstract— Software defect prediction is an important aspect of software development, as it helps developers and organizations to identify and resolve bugs in the software before they become major issues. In this paper, we explore the use of machine learning algorithms for software defect prediction. We discuss the different types of machine learning algorithms that have been used for software defect prediction and their advantages and disadvantages. We also provide a comprehensive review of recent studies that have used machine learning algorithms for software defect prediction. The paper concludes with a discussion of the challenges and opportunities in using machine learning algorithms for software defect prediction and the future directions of research in this field. This paper surveys the existing literature on software defect prediction, focusing specifically on deep learning techniques. Compared to existing surveys on the topic, this paper offers a more in-depth analysis of the strengths and weaknesses of deep learning approaches for software defect prediction. It explores the use of LSTMs for this task, which have not been extensively studied in previous surveys. Additionally, this paper provides a comprehensive review of recent research in the field, highlighting the most promising deep learning models and techniques for software defect prediction. The results of this survey demonstrate that LSTM-based deep learning models can outperform traditional machine learning approaches and achieve state-of-the-art results in software defect prediction. Furthermore, this paper provides insights into the challenges and limitations of deep learning approaches for software defect prediction, highlighting areas for future research and improvement. Overall, this paper offers a valuable resource for researchers and practitioners interested in using deep learning techniques for software defect prediction..

Keywords - Software defect prediction, Evaluation parameters, Long Short-Term Memory (LSTM), Recurrent neural network (RNN)

I. INTRODUCTION

Software defect prediction is an important aspect of software development, as it helps developers and organizations to identify and resolve bugs in the software before they become major issues. In this paper, we explore the use of machine learning algorithms for software defect prediction [30]. We discuss the different types of machine learning algorithms that have been used for software defect prediction and their advantages and disadvantages. We also provide a comprehensive review of recent studies that have used machine learning algorithms for software defect prediction [25]. The paper concludes with a discussion of the challenges and opportunities in using machine learning algorithms for software defect prediction and the future directions of research in this field. In this paper, we will first review the related work on software defect prediction and the evaluation parameters used by different methods. Then, we will describe the methodology used to train and evaluate the Long Short-Term Memory (LSTM) model for software defect prediction [34]. Next, we will present the experimental results and compare the performance of the LSTM model with other existing methods using various evaluation parameters [46]. Finally, we will discuss the limitations of our study and provide recommendations for future research in this area.

A. *Overview of machine learning algorithms for software defect prediction*

Software Defect Prediction can directly affect quality and has achieved significant popularity in last few years. Defective software modules have a massive impact over software's quality leading to cost overruns, delayed timelines and much higher maintenance costs. In this paper we have analyzed the most popular and widely used Machine Learning algorithms – Artificial Neural network, Support Vector Machine, Decision Tree, Association rule, Clustering [47]. The Primary concern of software development process is to ensure quality software at every development stage; therefore, a common goal and concern of each software development phase is to check and concentrate on improving the software quality. Software quality prediction thus aims to evaluate software quality level periodically and to indicate software quality problems early [1]. Commonly it is also called as a fault (bug) between software experts [2]. It is not so easy to manage quality software because of raising difficulties and several restrictions under which software is developed. Conversely, the software development organizations are not ready to take much risk with delivering inferior quality software [3]. Moreover, it leads to disappointment among customers.

B. The role of Long Short-Term Memory (LSTM) in software defect prediction

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series [60]. LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare [15]. As our dataset will have real time implementation and have different software development logics which will comprise of different logical defects and will be good for CPDP dynamic datasets [56]. Therefore, LSTM is the most suitable methodology for our research and will also reduce the research gaps of many problems coming across for SDP Model across dynamic systems which are evolved over time [37].

To capture the long-term dependencies that often exist between code elements we use a prediction system on a deep learning LSTM (Long Short-Term Memory) network [23]. We evaluate with a dataset from opensource Java project, namely from the Apache Project Repository. From the results of this study, it was concluded that the results of training using the LSTM network were both done by restarting and sequentially got higher accuracy, precision, recall, and f1-score results than using the RNN algorithm. The highest accuracy value is obtained by 93% using LSTM then precision is obtained at 89%, recall is 92% and f1-score is 90%. [71] From the above researches and surveys LSTM is one of the most appropriate methodologies to implement SDP Model Using HCPDP for dynamic datasets software testing [49].

Most of the studies available in literature have used historical data related to the same projects for identification of faulty modules however, availability of historical data for new software projects is not possible [20]. In case of new software projects, data for defect prediction is taken from similar types of projects developed earlier and this technique of defect prediction is called cross project defect prediction. From the past studies and applicability of hybrid search-based algorithms for cross project defect prediction is investigated [63]. Performance of hybrid search-based algorithms had been compared for with-in and cross project defect prediction. Hybrid search-based algorithms combine the advantages of search-based algorithms with machine learning techniques [18].

Survey Results showed that hybrid search-based algorithms are more suitable in case of cross project defect prediction in comparison to with-in project defect prediction [7]. Existing CPDP methods are based on the assumption that source and target projects should have the same metrics. Heterogeneous cross-project defect prediction (HCPDP) builds a prediction model using heterogeneous source and target projects. Existing HCPDP methods just focus on one source project or multiple source projects with the same metrics. These methods limit the scope of getting the source project [5].

Therefore, we propose Heterogeneous Defect Prediction with Multiple source projects (HDPM) which can use multiple heterogeneous source projects for defect prediction. HDPM based on transfer learning which can learn knowledge from one domain and use it to help with another domain. HDPM constructs a projective matrix between heterogeneous source and target projects to make the distributions of source and target projects similar [68].

II. LITERATURE REVIEW

Software defect is an error, bug, flaw, fault, malfunction or mistakes in software that causes it to create an erroneous or unpredicted outcome. Faults are essential properties of a system [6]. They appear from design or manufacture, or external environment. The majorities of the faults are from source code or design, some of them are from the incorrect code generating from compilers. Software Defect Prediction [SDP] plays an important role in the active research areas of software engineering [9]. The major risk factors related with a software defect which is not detected during the early phase of software development are time, quality, cost, effort and wastage of resources. Thus, the key objective of any organization is to determine and correct the defects in an early phase of Software Development Life Cycle [SDLC]. To improve the quality of software, datamining or machine learning techniques can be been applied to build predictions regarding the failure of software components by exploiting past data of software components and their defects. The main objective of software defect prediction is to improve the quality, minimized cost and time of software products. Software defect is also referred to as bug can be defined as shortage in the software product that causes the software not to perform its task as the programmer and customer needed [12].

Machine Learning is one of the most vital and motivating area of research with the objective of finding meaningful information from huge data sets [7]. The basic purpose of machine learning is to extract useful pattern from the data, mining data may be structured format (example. multiple data base) or text mining: unstructured data (example, natural language document). The main aim of software defect management is to amplify the quality of software by identifying and fixing the defects in the early phase of SDLC. The various phases of SDLC are requirements gathering phase, analysis phase, designing phase, coding phase, testing phase, implementation and maintenance phase. SDP plays a vital role in developing high quality software [8]. Identifying the defects in a preliminary stage of a SDLC is a very complicated job, hence efficient methods to be applied in order to remove them [23].

Software bugs are classified into three types:

- Software Defects by its Nature
- Software Defects by its Severity
- Software Defects by its Priority

Out of all the 3 types of defects our main area of research will be Software Defects by its Nature because this type of error mostly affects the software on the large scale and their output can give us unpredictable results [10].

In software defect prediction, the priority of errors refers to the order in which defects should be addressed. This can be determined by several factors, such as the severity of the error, the likelihood of it being exploited, and the impact it has on the system or users. Some common prioritization methods include:

- Severity-based: Errors are prioritized based on their potential impact, with critical errors being

***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

addressed first and less severe errors being addressed later.

- Risk-based: Errors are prioritized based on the likelihood of them being exploited, with high-risk errors being addressed first and low-risk errors being addressed later.
- Impact-based: Errors are prioritized based on the impact they have on the system or users, with errors that have a high impact being addressed first and those with a low impact being addressed later.

In this experiment, we have used 3 open source publicly available data from PROMISE Software Engineering Database. These datasets Tim Menzies et al. have been used in their research paper [58]. In another study, Jureczko et al. [59] have been assembled a software fault prediction model to predict the software defects using machine learning algorithms. They have discussed in their paper about 8 projects (PROMISE Repository) data and by taking 19 CK metrics and McCabe metrics for constructed a predictive model. In our study, we have used 22 attributes for building our automated fault predict model. Table 13 shows 22 different attributes from software defect datasets including 21 independent metrics and one is outcome information i.e., is faulty and no-fault.

TABLE I. LIST OF SOFTWARE METRICS [2]

No	Metrics Name	Type
1	Line of Code	McCabe
2	Cyclomatic complexity	McCabe
3	Essential complexity	McCabe
4	Design complexity	McCabe
5	Halstead operators and Operands	Halstead
6	Halstead volume	Halstead
7	Halstead program length	Halstead
8	Halstead difficulty	Halstead
9	Halstead intelligence	Halstead
10	Halstead effort	Halstead
11	Halstead time estimator	Halstead
12	Halstead line count	Halstead
13	Halstead comments count	Halstead
14	Halstead blank line count	Halstead
15	IO code and comments	Miscellaneous
16	Unique operators	Miscellaneous
17	Unique operands	Miscellaneous
18	Total operators	Miscellaneous
19	Total operands	Miscellaneous
20	Branch count	Miscellaneous
21	b: numeric	Halstead
22	Defects	False or True

The present study used JM1, CM1, PC1 datasets which were implemented in C language. Table 14 depicted details about detail of all datasets with their features.

TABLE II. DATASET DESCRIPTION [2]

Dataset	Missing Attributes	Instance	Class Distribution	
			True	False
JM1	None	10885	8779 (80.65%)	2106 (19.35%)
CM1	None	498	49 (9.83%)	449 (90.16%)
PC1	None	1109	1032 (93.05%)	77 (6.94%)

Software defect prediction is an important task in software engineering, and various datasets have been used to evaluate the performance of different prediction models. Three commonly used datasets are CM1, JM1, and PC1.

CM1 is a dataset containing data from the NASA MDP software development project. It consists of 498 instances, with each instance containing 21 features related to the code complexity, size, and structure. The dataset is binary, with each instance labelled as either defective or non-defective. The CM1 dataset has been widely used in research on software defect prediction, and several machine learning and statistical models have been evaluated using this dataset.

JM1 is another dataset commonly used for software defect prediction, consisting of data from the NASA software development project. It contains 10885 instances, with each instance containing 22 features related to the code complexity, size, and structure. Similar to CM1, the dataset is binary, with each instance labeled as either defective or non-defective. The JM1 dataset has also been used extensively in research on software defect prediction, and various machine learning models have been evaluated using this dataset [45].

PC1 is a dataset containing data from a large industrial software development project. It consists of 1109 instances, with each instance containing 22 features related to the code complexity, size, and structure. Unlike CM1 and JM1, the PC1 dataset is multi-class, with each instance labeled as either non-defective or one of six different types of defects. The PC1 dataset has been used to evaluate the performance of various machine learning and statistical models for multi-class software defect prediction [56].

Overall, these datasets have been widely used in research on software defect prediction and have helped researchers to develop and evaluate different prediction models. However, it is important to note that these datasets have limitations and may not fully represent the diversity of software development projects. Therefore, it is essential to use multiple datasets and perform cross-dataset evaluations to ensure the generalizability of software defect prediction models [57].

A. *Long Short Term Memory Algorithm*

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series [60]. LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare. As our dataset will have real time implementation and have different software development logics which will comprise of different logical defects and will be good for CPDP dynamic datasets [67]. Therefore, LSTM is the most suitable methodology for our research and will also reduce the research gas of many problems coming across for SDP Model across dynamic systems which are evolved over time [34].

Compared with traditional SDP methods, DP-LSTM uses the rich semantic and contextual information from the AST, which can be hardly represented by the handcrafted features [52]. Such information indicates how the codes are organized and interact with each other. Like many deep learning-based methods, DP-LSTM can seize these important features which are then used to train the classifier. [70] Compared with other deep learning-based models (e.g., CNN, DBN), DP-LSTM can learn the relationship of longer dependencies. Since defective codes are closely related to its context, but if the context becomes further, models like CNN, which focus on local features, may fail to capture the connection of the codes and its context. Also, the bidirectional structure of LSTM ensures that both previous and subsequent code segments are taken into consideration. Although CNN can generate semantic and contextual features, we believe that features generated by DP-LSTM can better represent how the defect is caused [70].

Algorithm like LSTM which is well known for Long-Short Term Memory in ML and another one is hybrid SVM based decision tree [24]. These algorithms are because they have their own way to handle the real time datasets and as discussed earlier, we are in the way to get real time software from the different software companies. Dynamic software dataset has different Software Development Strategies [25]. Therefore, we require the methodology which can help us out with high storage of memory consumption.

The paper "On the use of deep learning in software defect prediction" by Giray et al. (2022) explores the effectiveness of deep learning techniques in software defect prediction. The study evaluates the performance of various deep learning models, such as convolutional neural networks and recurrent neural networks, on benchmark datasets and compares them to traditional machine learning methods [73].

In the paper "A comparative study on the effect of data imbalance on software defect prediction" by Liu et al. (2022), the authors investigate the impact of imbalanced data on the accuracy of software defect prediction models. The study compares the performance of various machine learning algorithms on imbalanced datasets and highlights the importance of a balanced dataset for accurate defect prediction [74]. The paper "Ensemble Machine Learning Paradigms in Software Defect Prediction" by Sharma et al. (2023) proposes the use of ensemble learning techniques in software defect prediction. The study compares the performance of various ensemble approaches, such as bagging and boosting, on benchmark datasets and highlights the potential of ensemble learning for improving defect prediction accuracy [75].

From the past literature review and study of relevant SDP Software we have prompted over dedication for Cross Project Defect Prediction [CPDP]. Our main focus of area will be there may not be enough historical data can apply to the prediction model. A possible solution to this problem is cross-project defect prediction, that is, using other project data to build prediction models. The increasing number of open datasets in various fields has attracted more researchers' attention and promoted more practice in cross-project defect prediction. Our next study of area is for Cross Project Defect Prediction Similar dataset and heterogenous dataset.

B. Comparison of LSTM with other Machine Learning Algorithms

Machine Learning Algorithms for Software Defect Prediction: There are several types of machine learning algorithms that have been used for software defect prediction, including decision trees, support vector machines (SVM), neural networks, and Bayesian networks [35]. These algorithms have been used to predict software defects based on different types of data, including code metrics, historical data, and software artifacts [61]. Decision trees are a popular machine learning algorithm for software defect prediction. Decision trees are simple, easy to understand, and interpretable. They are used to model the relationship between input variables and the output variables. Decision trees have been used for software defect prediction by using code metrics, such as lines of code, cyclomatic complexity, and number of comments, as input variables. Decision trees have been found to be effective for software defect prediction and have been used in many studies [47].

Support Vector Machines (SVM): Support vector machines (SVM) are another type of machine learning algorithm that have been used for software defect prediction. SVM is a supervised learning algorithm that can be used for both classification and regression. SVM has been used for software defect prediction by using code metrics and historical data as input variables. SVM has been found to be effective for software defect prediction and has been used in many studies [22].

Neural Networks: Neural networks are a type of machine learning algorithm that are inspired by the structure and function of the human brain. Neural networks have been used for software defect prediction by using code metrics and historical data as input variables. Neural networks have been found to be effective for software defect prediction and have been used in many studies [22].

Bayesian Networks: Bayesian networks are a type of probabilistic graphical model that represents the relationships between variables. Bayesian networks have been used for software defect prediction by using code metrics and historical data as input variables. Bayesian networks have been found to be effective for software defect prediction and have been used in many studies. There have been many recent studies that have used machine learning algorithms for software defect prediction. These studies have used different machine learning algorithms and have used different types of data as input variables. Some of the recent studies have found that machine learning algorithms are effective for software defect prediction, while others have found that machine learning algorithms have limitations.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

TABLE III. LITERATURE REVIEW

Sr no.	Paper Title	Publication Year	Journal Name	Publication	Methodology	Research Gap
1	On the use of deep learning in software defect prediction	2022	The Journal of Systems & Software	ScienceDirect	Deep Learning	The study focuses on evaluating the effectiveness of deep learning techniques in software defect prediction, which can be used as an alternative to traditional machine learning methods.
2	A comparative study on the effect of data imbalance on software defect prediction	2022	Procedia Computer Science	ScienceDirect	Comparative Study	The study compares the performance of different machine learning algorithms on imbalanced software defect datasets and highlights the need for a balanced dataset to achieve better accuracy in defect prediction.
3	Ensemble Machine Learning Paradigms in Software Defect Prediction	2023	Procedia Computer Science	ScienceDirect	Ensemble Learning	The study proposes the use of ensemble learning techniques in software defect prediction and compares the performance of different ensemble approaches. The research highlights the potential of ensemble learning for improving defect prediction accuracy.
3.	A tool for creating datasets and software defect predictions	2022	ScienceDirect	Elsevier	Java Swing, REST API, GitHub, SQLite, Relational database, Open JDK	End-to-end machine learning predictions and multi-label defect predictions no supported
4.	Performance of Heterogeneous Ensemble Approach With Traditional Methods Based on Software Defect Detection Model	2022	Journal of Theoretical and Applied Information Technology	Little Lion Scientific	SVM, ANN, Random Forest	Used only PROMISE Software Engineering repository and no dynamic dataset is used
5.	An Attribute Selection Process for Cross-Project Software Defect Prediction	2021	Research gate		Within project and cross-project domain using NASA MDP repository	Meta-heuristic approaches not adopted
6.	Towards Design and Feasibility Analysis of DePaaS: AI Based Global Unified Software Defect Prediction Framework	2022	MDPI	Applied Sciences	Artificial Intelligence, DePaaS	Larger dataset coverage is not done.
7.	Machine Learning-Based Software Defect Prediction for Mobile Applications: A Systematic Literature Review	2022	MDPI	Sensors	ML Algorithms: LSTM. DBN, DNN	unsupervised and semi-supervised learning for mobile defect prediction.

Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

8.	Defect Prediction Using Akaike and Bayesian Information Criterion	2021	Computer Systems Science & Engineering	Tech Science Press	ANN - Akaike information criterion (AIC) and the Bayesian information criterion (BIC)	Heterogenous CPDP not taken in consideration gap in research observed
9.	Interpretability application of the Just-in-Time software defect prediction model	2022	The Journal of Systems & Software	Elsevier	Random Forest Classification, LIME model	no studies on defect types and locations
10.	A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools	2022	Engineering Applications of Artificial Intelligence	Elsevier	Artificial Intelligence techniques	Industry adoption of Software Defect Prediction
11.	Software Defect Prediction using Machine Learning Algorithms: Current State of the Art	2021	Scopus	Solid State Technology	SVM, Random Forest, Decision Tree	Use of hybrid OO metrics in the machine learning dimension
12.	A Novel Cross-Project Software Defect Prediction Algorithm Based on Transfer Learning	2022	IEEE Explore	Tsinghua Science and Technology	Transfer-leaning algorithm (TSboostDF)	Multi-source transfer learning on CPDP
13.	Cross-project defect prediction based on G-LSTM model	2022	Elsevier	Pattern Recognition Letters	LSTM	Method outperforms some traditional and state-of-the-art CPDP methods in terms of the evaluation metrics of AUC and Acc.

TABLE IV. COMPARISON OF LSTM WITH OTHER MACHINE LEARNING ALGORITHMS FOR SOFTWARE DEFECT PREDICTION

Sr no.	Algorithm	Acceptance	Avoidance
1.	RNN [41]	Short Term Dependencies – good for short term memories	Long Dependencies not supported (making predictions for present) Vanishing Gradient and exploding gradient makes it unusable.
2.	LSTM [26]	Long Short-Term Memory – Cell Unit inserted updates in every loop – considers current input, previous output and memory	LSTMs take longer to train. LSTMs require more memory to train.
3.	CNN [43]	Good for image processing algorithm	CNN has several layers then the training process takes a lot of time if the computer doesn't consist of a good GPU
4.	ANN [48]	ANN can handle more than one task at the same time.	ANN need processors that support parallel processing, so the ANNs are dependent on the hardware.
5.	SVM [28]	Effective in high dimensional cases	SVM algorithm is not suitable for large data sets
6.	LR [12]	Updated easily to reflect new data	On high dimensional datasets, this may lead to the model being over-fit on the training set
7.	Random forest [12]	It can handle the data set containing continuous variables - performs better results for classification problems.	Random Forest algorithm may change considerably by a small change in the data.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

III. METHODOLOGY

HCPDP stands for the "High Confidence Software Defect Prediction" dataset. It is a publicly available dataset that is used to train and evaluate machine learning models for software defect prediction. The dataset contains information on code changes and corresponding defect labels for several open-source software projects. The dataset is curated by researchers and it's aimed to provide a benchmark for software defect prediction models. The dataset provides information such as the number of lines added, deleted, modified and the number of files changed, and whether the change led to a defect or not. This dataset is commonly used to test the performance of software defect prediction models, and can be used to train and evaluate machine learning models, such as LSTM, to predict the likelihood of a defect being introduced by a new code change.

Heterogeneous cross-project defect prediction (HCP) is a method for using data from one software project to predict defects in a different, but related, software project. This is in contrast to traditional cross-project defect prediction, which uses data from multiple projects within the same domain or organization. The goal of HCP is to leverage the information from one or multiple projects to improve the defect prediction performance in a target project. It can be useful in scenarios where there is limited data available for the target project, but more data is available for related projects.

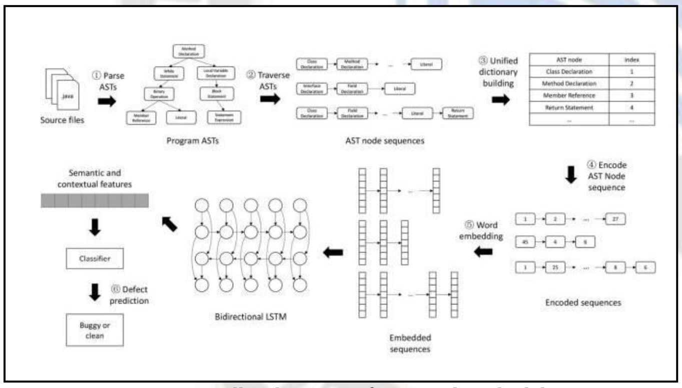

Figure 1. Overall architecture of Proposed Methodology

HCP approaches typically involve training a defect prediction model on data from one or more source projects, and then applying the trained model to the target project to make predictions. This process often involves pre-processing the data from the source and target projects to align them, and then using techniques such as transfer learning, ensemble learning or meta-learning to improve the performance of the model on the target project.

A. *Model Development and Evaluation*

The steps for using an LSTM algorithm for software defect prediction are:

1. Collect and pre-process the data: Gather historical data on code changes, bug reports, and other relevant information. Pre-process the data to format it for use with an LSTM model.
2. Split the data into training and testing sets: Divide the data into a training set, which will be used to train the model, and a testing set, which will be used to evaluate the model's performance.
3. Train the LSTM model: Use the training data to train the LSTM model. This typically involves specifying the architecture of the LSTM network, such as the number of layers and units, and then training the model using a set of training data.
4. Evaluate the model: Use the testing data to evaluate the model's performance in making defect predictions.
5. Fine-tune the model: Based on the results of the evaluation, make adjustments to the model, such as changing the architecture or adjusting the training parameters, to improve its performance.
6. Use the model to make predictions: Once the model is trained and fine-tuned, it can be used to make predictions on new code changes.
7. Monitor the model's performance over time: As new data becomes available, retrain the model and monitor its performance to ensure it continues to make accurate predictions

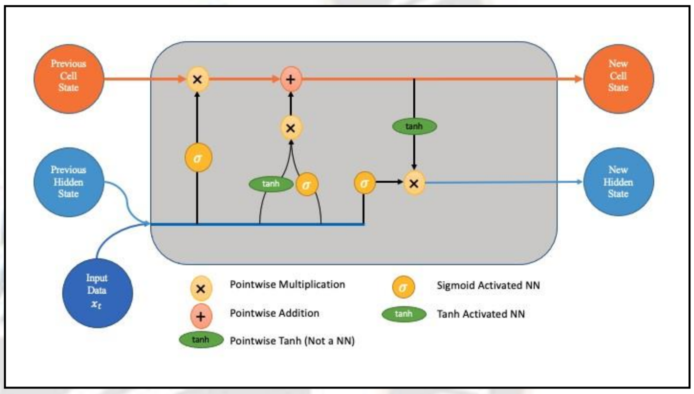

Figure 2. Long Short-Term Memory Architecture

IV. RESULTS AND DISCUSSION

In this experiment, 6 machine learning (ML) techniques have been considered to construct the defect model: Decision Tree (DT), k- nearest neighbors (KNN), Logistics Regression (LR), Naïve Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM) [37]. Below given is the prediction accuracy and F1 Value of the different algorithms performed using different dataset values. Especially, defective modules are very crucial than not faulty modules. In our experiment, we used 10-fold cross-validation technique to evaluate the performance of six classification techniques. To determine the parameters for the software defect model, we used the different data preprocessing methods that have been increased the accuracy and consistency of the classification model. Table 4 shows the performance evaluation of six supervised classification techniques for software fault prediction. With respect to the precision: DT and SVM achieved the highest performance (i.e., 100%) on JM1 datasets; DT, NB, SVM, and RF achieved the best performance on CM1 datasets, (it's respectively 100%); DT, SVM, and RF obtained the highest performance (i.e., 97%) on PC1 datasets. Relatively, all of the classifiers have shown good performance in terms of precision. However, considering the recall of the analysis, SVM and RF achieved the highest performance on JM1 datasets; LR and NB attained the lowest performance on CM1

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

and PC1 datasets. Not that all of the classifiers achieved very similar scores in terms of recall. Another measure for classification is F1 measure. With respect to F1 measure: SVM achieved the highest value (i.e., 100%) on JM1 datasets and NB obtained the lowest score (i.e., 93%). By Looking CM1 datasets, we can monitor that the f1 scores are mostly similar (i.e., NB, DT, SVM, RF = 100% and KNN = 97%, LR = 95%). Moreover, RF achieved the best score (i.e., 99%) and KNN performed lowest (86%) on PC1 datasets. In addition, all of the classifiers have achieved utmost performance on JM1, CM1, and PC1 datasets, in terms of accuracy. This indicates that all of the classifiers are very effective in their classification performance to predict software defect modules.

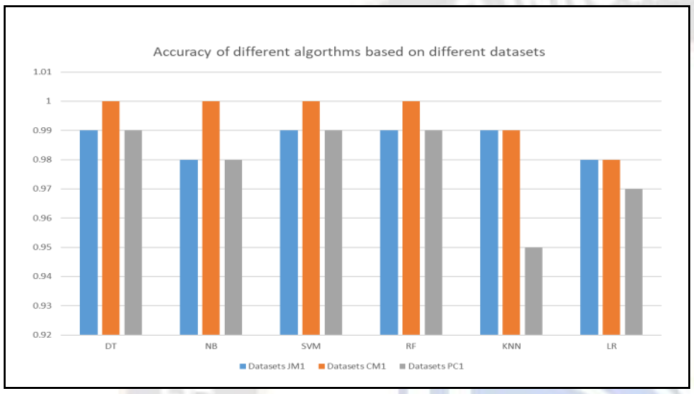

Figure 3. Overall comparison of different algorithms based on different datasets

In a software defect prediction model using machine learning, the first priority should be given to errors that have the highest potential impact on the system or users. This can be determined by several factors, such as the severity of the error, the likelihood of it being exploited, and the impact it has on the system or users [35]. One common approach to prioritizing errors in a software defect prediction model using machine learning is to use a combination of several metrics, such as:

- **Error rate:** The number of errors predicted by the model divided by the total number of instances in the dataset
- **Precision:** The number of true positive predictions divided by the total number of positive predictions
- **Recall:** The number of true positive predictions divided by the total number of actual positive instances in the dataset
- **F1 score:** The harmonic means of precision and recall

Errors that have high error rate and F1 score, and low precision and recall are considered as high priority errors to be addressed as soon as possible.

Confusion matrix is mainly used to show & evaluate the performance of a certain classification model in which we know what are the real positive values that are true among the data set.

TP: true positive, where the predicted output is the same as the actual one (both are positive).

FP: false positive, where the predicted output is positive while the actual is negative.
FN: false negative, where the predicted output is negative while the actual is positive.
TN: true negative, where the predicted output is the same as the actual one (both are negative).

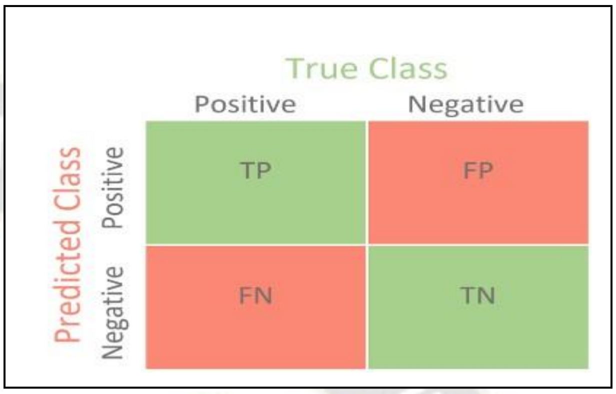

Figure 4. Confusion matrix [46]

Accuracy: Accuracy is the most widely used evaluation parameter for software defect prediction. It measures the proportion of correctly predicted instances among all the instances. However, accuracy alone may not be sufficient as it does not consider the imbalance between the number of defective and non-defective instances.

Accuracy with high value is so important in totally showing that our model is perfectly working, but only in datasets having the same FP and FN (symmetric), if our dataset isn't of that type, here, other parameters should be taken into consideration in the evaluation process.

Accuracy = TP+TN/TP+FP+FN+TN (1)

Precision: Precision is the number of true positives divided by the sum of true positives and false positives. It measures the proportion of correctly predicted defects among all predicted defects. Precision is useful when the cost of false positives is high.

Precision = TP/TP+FP (2)

Recall: Recall is the number of true positives divided by the sum of true positives and false negatives. It measures the proportion of correctly predicted defects among all actual defects. Recall is useful when the cost of false negatives is high.

Recall = TP/TP+FN (3)

F1 score: F1 Score is the weighted average of Precision and Recall. This score mainly takes into consideration both FP and FN. An F1 score is considered perfect when it’s 1, while the model is a total failure when it’s 0. In some cases, F1 is more preferable and useful than accuracy, especially if you have an uneven class distribution or a biased distribution. When having a similar cost in comparison between FP and FN, here accuracy performs better. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)   (4)

Receiver Operating Characteristic (ROC) curve: ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. It helps to evaluate the performance of a classifier at different levels of sensitivity and specificity.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) that has been used for software defect prediction. LSTM has been found to be effective in capturing the temporal dependencies in software code. LSTM can be evaluated using some of the evaluation parameters discussed above, such as accuracy, precision, recall, and F-measure. However, ROC curve may not be applicable to LSTM as it does not produce probabilistic predictions. Instead, a threshold can be set on the output of the LSTM to convert it into a binary prediction, and then the above evaluation parameters can be used.

V. Conclusion

Software defect prediction is an important aspect of software development and the use of machine learning algorithms for software defect prediction has been growing in recent years. Despite some challenges, there are many opportunities for machine learning algorithms to improve the accuracy and efficiency of software defect prediction. As the field of machine learning continues to evolve, it is likely that machine learning algorithms will play an increasingly important role in software development. In this experimental study, we proposed an automated software engineering approaches for defect prediction model development (SDPD) on software development life cycle. After that, the main objective of our study was to evaluate the abilities of six supervised based the machine learning classifications techniques to predict the software defect modules using 3 NASA datasets. The results (i.e., accuracy: 98-100%) of the experiment with different attributes showed the capability and efficiency of the SDPD model to identify the fault and improve software quality. In addition, this SDPD model can be able to early detection of software faults by collecting real-time software development data from the target applications. The proposed approach can be used for software fault recovery inside a system and enhanced by applying machine learning techniques to construct SDPD more effective in software fault retrieval. For future work, we will implement more classification algorithms, such as hybrid or ensemble model to verify the performance of the software fault prediction.

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression, "One of us (R.B.G.) thanks . . ." Instead, try "R.B.G. thanks". Put applicable sponsor acknowledgments here; DO NOT place them on the first page of your paper or as a footnote.

References

[1] Xing, F. , Guo, P. , Lyu, M. R. "A Novel Method for Early Software Quality Prediction Based on Support Vector Machine". 2005,In: Proceedings of The 16th IEEE International Symposium on Software Reliability Engineering

[2] Ahmed, Md. Razu & Ali, Md. Asraf & Ahmed, Nasim & Zamal, Md Fahad & Shamrat, F M. (2020). The Impact of Software Fault Prediction in Real-World Application: An Automated Approach for Software Engineering. 10.1145/3379247.3379278.

[3] Emam, K.,El., "The ROI from Software Quality". Auerbach Publications, Taylor and Francis Group, LLC, (2005).

[4] Khoshgoftaar, T.M., Allen, E.B., Kalaichelvan, K.S., Goel, N., "Early Quality Prediction: A Case Study in Telecommunications".2006, IEEE Software.

[5] M. Jureczko, "Significance of different software metrics in defect prediction", Software Engineering: An International Journal, 1.1, 2011, pp. 86-95.

[6] S.S. Rathore, A. Gupta, "Investigating object-oriented design metrics to predict fault-proneness of software modules", In Software Engineering (CONSEG), CSI Sixth International Conference: IEEE, 2012, pp. 1-10.

[7] M.M. Rosli, N.H.I. Teo, N.S.M. Yusop, N.S. Mohammad, "The design of a software fault prone application using evolutionary algorithm", In Open Systems (ICOS), IEEE, 2011, pp. 338-343.

[8] M. Jureczko, L. Madeyski, "Towards identifying software project clusters with regard to defect prediction", In Proceedings of the 6th International Conference on Predictive Models in Software Engineering:ACM, 2010, page 9

[9] W. Afzal, R. Torkar, R. Feldt, "Prediction of fault count data using genetic programming.", In Multitopic Conference, INMIC: IEEE International, 2008, pp. 349-356.

[10] X.Y.Jing, S. Ying, Z.W. Zhang, S.S. Wu, J. Liu, "Dictionary learning based software defect prediction", In Proceedings of the 36th International Conference on Software Engineering, ACM, 2014, pp. 414-423

[11] X.Y. Jing, Z.W. Zhang, S. Ying, F. Wang, Y.P.Zhu, "Software defect prediction based on collaborative representation classification", In Companion Proceedings of the 36th International Conference on Software Engineering: ACM, 2014, pp. 632-633.

[12] R. Verma, A. Gupta, "Software defect prediction using Two level data pre-processing", In Recent Advances in Computing and Software Systems (RACSS), International Conference:IEEE, 2012, pp. 311-317.

[13] Li, Zhang, R.Wu, H.Zhou, "Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engineering", Vol.19, No.2, 2012, pp. 201-230.

[14] Song, Q., Jia, Z., Shepperd, M., Ying, S., Liu, J,"A general software defect-proneness prediction framework", IEEE Transactions on Software Engineering, Vol.37, No.3, 2011, pp. 356-370.

[15] Marek Leszak, Dewayne E. Perry, Dieter Stoll, "A Case Study in Root Cause Defect Analysis", ICSE, 2000.

[16] N.Kalaivani, Dr.R.Beena, "Overview of Software Defect Prediction using Machine Learning Algorithms", International Journal of Pure and Applied Mathematics, 2018.

[17] Ren Jinsheng, Qin Ke, "On Software Defect Prediction Using Machine Learning", Journal of Applied Mathematics, 2014.

[18] Prabha C.Lakshmi, Shivakumar N., "Software Defect Prediction Using Machine Learning Techniques", 4th International Conference on Trends in Electronics and Informatics (ICOEI), 2020.

[19] Marwa, Assim, Obeidat Qasem, "Software Defects Prediction using Machine Learning Algorithms", International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020.

[20] Martin Shepperd, David Bowes and Tracy Hall, "Researcher Bias: The Use of Machine Learning in Software Defect Prediction", IEEE Transactions on Software Engineering, 2014.

[21] Laila Bergmane, Jānis Grabis, Edžus Žeiris, "A Case Study: Software Defect Root Causes", Information Technology and Management Science, December 2017.

[22] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah, "Software Bug Prediction using Machine Learning Approach", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 2, 2018.

[23] Pooja Paramshetti, D. A. Phalke, "Survey on Software Defect Prediction Using Machine Learning Techniques", International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064, 2012.

[24] Altexsoft, "Comparing Automated Testing Tools: Selenium, TestComplete, Ranorex, and more", Feb, 2018, https://www.altexsoft.com/blog/engineering/comparing-automated-testing-tools-selenium-testcomplete-ranorex-and-more/

[25] Katalon, "A Comparison of Automated Testing Tools", 2020, https://www.katalon.com/resources-center/blog/comparison-automated-testing-tools/

[26] Kanade, A.; Maniatis, P.; Balakrishnan, G.; Shi, K. Learning and Evaluating Contextual Embedding of Source Code. In Proceedings of the 37th International Conference on Machine Learning; Daumé , H., III, Singh, A., Eds.; PMLR: 2020; Volume 119, pp. 5110–5121. Available online: http://proceedings.mlr.press/v119/kanade20a.html (accessed on 17 December 2020).

[27] Raychev, V.; Bielik, P.; Vechev, M. Probabilistic Model for Code with Decision Trees. SIGPLAN Not. 2016, 51, 731–747.

[28] Raychev, V.; Bielik, P.; Vechev, M.; Krause, A. Learning Programs from Noisy Data. SIGPLAN Not. 2016, 51, 761–774.

[29] Alon, U.; Brody, S.; Levy, O.; Yahav, E. code2seq: Generating Sequences from Structured Representations of Code. arXiv 2019, arXiv:cs.LG/1808.01400.

[30] Allamanis, M.; Sutton, C. Mining source code repositories at massive scale using language modeling. In Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, USA, 18–19 May 2013; pp. 207–216.

[31] Iyer, S.; Konstas, I.; Cheung, A.; Zettlemoyer, L. Summarizing source code using a neural attention model. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 7–12 August 2016; pp. 2073–2083.

[32] Bryksin, T.; Petukhov, V.; Alexin, I.; Prikhodko, S.; Shpilman, A.; Kovalenko, V.; Povarov, N. Using Large-Scale Anomaly Detection on Code to Improve Kotlin Compiler. In Proceedings of the 17th International Conference on Mining Software Repositories, MSR '20, Seoul, Korea, 29–30 June 2020; pp. 455–465.

[33] Allamanis, M.; Brockschmidt, M.; Khademi, M. Learning to Represent Programs with Graphs. arXiv 2018, arXiv:cs.LG/1711.00740.

[34] Mauša, G.; Galinac-Grbac, T.; Dalbelo-Baši´c, B. A systematic data collection procedure for software defect prediction. Comput. Sci. Inf. Syst. 2016, 13, 173–197.

[35] Sayyad Shirabad, J.; Menzies, T. The PROMISE Repository of Software Engineering Databases; School of Information Technology and Engineering, University of Ottawa: Ottawa, ON, Canada, 2005. Available online: http://promise.site.uottawa.ca/SERepository/ (accessed on 17 December 2020).

[36] Shepperd, M.; Song, Q.; Sun, Z.; Mair, C. NASA MDP Software Defects Data Sets. 2018. Available online: https://figshare.com/collections/NASA_MDP_Software_Defects_Data_Sets/4054940/1 (accessed on 17 December 2020).

[37] Afric, P.; Sikic, L.; Kurdija, A.S.; Silic, M. REPD: Source code defect prediction as anomaly detection. J. Syst. Softw. 2020, 168, 110641.

[38] Xu, J.; Wang, F.; Ai, J. Defect Prediction With Semantics and Context Features of Codes Based on Graph Representation Learning. IEEE Trans. Reliab. 2020, 1–13.

[39] Ferenc, R.; Gyimesi, P.; Gyimesi, G.; Tóth, Z.; Gyimóthy, T. An automatically created novel bug dataset and its validation in bug prediction. J. Syst. Softw. 2020, 169, 110691.

[40] Tóth, Z.; Gyimesi, P.; Ferenc, R. A Public Bug Database of GitHub Projects and Its Application in Bug Prediction. In Proceedings of the Computational Science and Its Applications—ICCSA, Beijing, China, 4–7 July 2016; Springer International Publishing: Cham, Switzerland, 2016; pp. 625–638.

[41] Ferenc, R.; Tóth, Z.; Ladányi, G.; Siket, I.; Gyimóthy, T. A public unified bug dataset for java and its assessment regarding metrics and bug prediction. Softw. Qual. J. 2020, 28, 1447–1506.

[42] Tufano, M.; Watson, C.; Bavota, G.; Penta, M.D.; White, M.; Poshyvanyk, D. An Empirical Study on Learning Bug-Fixing Patches in the Wild via Neural Machine Translation. ACM Trans. Softw. Eng. Methodol. 2019, 28, 1–29.

[43] Widyasari, R.; Sim, S.Q.; Lok, C.; Qi, H.; Phan, J.; Tay, Q.; Tan, C.; Wee, F.; Tan, J.E.; Yieh, Y.; et al. BugsInPy: A database of existing bugs in Python programs to enable controlled testing and debugging studies. In Proceedings of the ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, 8–13 November 2020; Devanbu, P., Cohen, M.B., Zimmermann, T., Eds.; ACM: New York, NY, USA, 2020; pp. 1556–1560.

[44] Saha, R.K.; Lyu, Y.; Lam, W.; Yoshida, H.; Prasad, M.R. Bugs.Jar: A Large-Scale, Diverse Dataset of Real-World Java Bugs. In Proceedings of the 15th International Conference on Mining Software Repositories (MSR '18), Gothenburg, Sweden, 28–29 May 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 10–13.

[45] Jinyong Wang and Ce Zhang. 2018. Software reliability prediction using a deep learning model based on the RNN encoder-decoder. Reliab. Eng. Syst. Saf. (2018).

[46] Hoa Khanh Dam, Trang Pham, Shien Wee Ng, Truyen Tran, John Grundy, Aditya Ghose, Taeksu Kim, and Chul-Joo Kim. 2019. Lessons Learned from Using a Deep Tree-Based Model for Software Defect Prediction in Practice. In Proceedings of the 16th International Conference on Mining Software Repositories.

[47] Khanh Hoa Dam, Trang Pham, Shien Wee Ng, Truyen Tran, John Grundy, Aditya K. Ghose, Taeksu Kim, and Chul-Joo Kim. 2018. A deep tree-based model for software defect prediction. ArXiv (2018).

[48] J. Li, P. He, J. Zhu, and M. R. Lyu. 2017. Software Defect Prediction via Convolutional Neural Network. In IEEE International Conference on Software Quality, Reliability and Security (QRS).

[49] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional Neural Networks over Tree Structures for Programming Language Processing. In Proceedings of the Thirtieth AAAI Conference on Arti"cial Intelligence.

[50] Anh Phan, Le Nguyen, and Lam Bui. 2018. Convolutional Neural Networks over Control Flow Graphs for Software Defect Prediction. (2018).

[51] C. Manjula and Lilly Florence. 2019. Deep neural network based hybrid approach for software defect prediction using software metrics. Cluster Computing (2019).

[52] Haonan Tong, Bin Liu, and Shihai Wang. 2017. Software Defect Prediction Using Stacked Denoising Autoencoders and Two-stage Ensemble Learning. Information and Software Technology (2017).

[53] Song Wang, Taiyue Liu, and Lin Tan. 2016. Automatically Learning Semantic Features for Defect Prediction. In Proceedings of the 38th International Conference on Software Engineering.

[54] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E. Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. 2013. A Large-Scale Empirical Study of Just-in-Time Quality Assurance. IEEE Trans. Softw. Eng. (2013).

[55] Karim O. Elish and Mahmoud O. Elish. 2008. Predicting Defect-Prone Software Modules Using Support Vector Machines. J. Syst. Softw. (2008).

[56] N. Gayatri, Nickolas Savarimuthu, and A. Reddy. 2010. Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions. Lecture Notes in Engineering and Computer Science (2010).

[57] Taghi M. Khoshgoftaar and Naeem Seliya. 2002. Tree-Based Software Quality Estimation Models For Fault Prediction. In Proceedings of the 8th International Symposium on Software Metrics.

[58] Jun Wang, Beijun Shen, and Yuting Chen. [n.d.]. Compressed C4.5 Models for Software Defect Prediction. In Proceedings of the 2012, 12th International Conference on Quality Software.

[59] H. B. Bolat, G. T. Temur, and IGI Global, Agile approaches for successfully managing and executing projects in the fourth industrial revolution.

[60] G. P. Bhandari and R. Gupta, "Machine learning based software fault prediction utilizing source code metrics," in 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018, pp. 40–45.

[61] H. Tanwar and M. Kakkar, "A Review of Software Defect Prediction Models," Springer, Singapore, 2019, pp. 89–97.

[62] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Appl. Soft Comput., vol. 27, pp. 504–518, Feb. 2015.

[63] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," Inf. Softw. Technol., vol. 49, no. 5, pp. 483–492, May 2007.

[64] P. N. Tan, M. Steinbach, and V. Kumar, Introduction to data mining. Pearson Addison Wesley, 2005.

[65] Xuemei Peng "Research on Software Defect Prediction and Analysis Based on Machine Learning" 3rd International Conference on Modeling, Simulation (2022) 1742-6596.

[66] Jorayeva, M.; Akbulut, A.; Catal, C.; Mishra, A. Machine Learning-Based Software Defect Prediction for Mobile Applications: A Systematic Literature Review. Sensors (2022), 22, 2551

[67] Ponnala, Ramesh & REDDY . Software Defect Prediction using Machine Learning Algorithms: Current State of the Art. Solid State Technology-(2021) 64. 6541-6556.

[68] Rhmann, W. Cross project defect prediction using hybrid search based algorithms. Int. j. inf. tecnol. 12, 531–538 (2020). https://doi.org/10.1007/s41870-018-0244-7

[69] Yin XL, Liu L, Liu HX, Wu Q. Heterogeneous cross-project defect prediction with multiple source projects based on transfer learning. Math Biosci Eng. 2019 Nov 11;17(2):1020-1040. doi: 10.3934/mbe.2020054. PMID: 32233568.

[70] Vashisht, R., Rizvi, S.A.M. (2020). Heterogeneous Cross Project Defect Prediction – A Survey. In: Singh, P., Sood, S., Kumar, Y., Paprzycki, M., Pljonkin, A., Hong, WC. (eds) Futuristic Trends in Networks and Computing Technologies. FTNCT 2019. Communications in Computer and Information Science, vol 1206. Springer, Singapore. https://doi.org/10.1007/978-981-15-4451-4_22

[71] Software defect prediction via LSTM Jiehan Deng,Lu Lu,Shaojian Qiu First published: 01 August 2020 https://doi.org/10.1049/iet-sen.2019.0149

[72] Bahaweres, Rizal & Jumral, Detia & Hermadi, Irman & Suroso, Arif & Arkeman, Yandra. (2021). Hybrid Software Defect Prediction Based on LSTM (Long Short Term Memory) and Word Embedding. 70-75. 10.1109/ICON-SONICS53103.2021.9617182.

[73] Giray G., Bennin K. E., Köksal Ö., Babur Ö., & Tekinerdogan B. (2022). On the use of deep learning in software defect prediction. The Journal of Systems & Software, 184, 111280. https://doi.org/10.1016/j.jss.2021.111280

[74] Liu Y., Zhang W., Qin G., & Zhao J. (2022). A comparative study on the effect of data imbalance on software defect prediction. In 9th International Conference on Information Technology and Quantitative Management (pp. 1603-1616). ScienceDirect. Procedia Computer Science, 214. https://doi.org/10.1016/j.procs.2022.01.169

[75] Sharma T., Jatain A., Bhaskar S., & Pabreja K. (2023). Ensemble Machine Learning Paradigms in Software Defect Prediction. In International Conference on Machine Learning and Data Engineering (pp. 199-209). ScienceDirect. Procedia Computer Science, 218. https://doi.org/10.1016/j.procs.2022.12.028.