Mr. Raja Lodhi, 2024, 12:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

A Survey of Software Defect Prediction Using Machine Learning Algorithms

Research Scholar Mr. Raja Lodhi, Assistant Professor Rajkumar Sharma
Department of Computer Science & Engg.
Lakshmi Narain College of Technology, Bhopal

Abstract- Software Defect Prediction [SDP] plays an important role in the active research areas of software engineering. A software defect is an error, bug, flaw, fault, malfunction or mistake in software that causes it to create a wrong or unexpected outcome. 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. Defects may occur in any phase of software development. Booming software companies focus concentration on software quality, particularly during the early phase of the software development .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, data mining techniques have been applied to build predictions regarding the failure of software components by exploiting past data of software components and their defects. This paper reviewed the state of art in the field of software defect management and prediction, and offered data mining techniques in brief.

Keywords- Software defect prediction, data mining, machine leaning.

I. INTRODUCTION

1. Software Defect Prediction

Software Defect

A 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. They appear from design or manufacture, or external environment. Software flaws are programming errors which cause different performance compared with anticipation. The majorities of the faults are from source code or deign, some of them are from the incorrect code generating from compilers. For software developers and clients, software faults are a danger problem. Software defects not merely decrease software quality, increase costing but also delay the development schedule. Software fault predicting is proposed to solve this sort of trouble. SDP can efficiently progress the effectiveness of software testing and direct the allocation of resources. To develop quality software, software flaws have to be detected and corrected at early phase of SDLC.

Software Defect Management

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, and coding phase, testing phase, implementation and maintenance phase. SDP plays a vital role in developing high quality software. 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.

The main stages in defect handling include [1]:

- Identifying the defects
- Categorizing the defects
- Analyzing the defects
- Predicting the defects
- Removing the defects

The first step is to identify the occurrence of defects in software. Code inspection, building a prototyping model and testing are used to identify the defects in software. After identifying the defects, the defects should be categorized, analyzed, predicted and detected.

Software Defect Prediction [SDP]

SDP identifies the module that are defective and it requires wide range of testing. Early identification of an error leads to effective allocation of resources, reduces the time and cost of developing a software and high quality software. Therefore, an SDP model plays a vital role in understanding, evaluating and improving the quality of a software system.

II. LITERATURE REVIEW

Peng He et al. conducted an empirical study on software defect prediction with a simplified metric set [2]. Research has been conducted on 34 releases of 10 open source projects available at PROMISE repository. The finding indicates the result of top-k metrics or minimum metric subset provides acceptable output compared with benchmark predictors. The simplified or minimum mertic set works well in case of minimum resources.

Grishma BR et al. investigated root cause for fault prediction by applying clustering techniques and identifies the defects occurs in various phases of SDLC.

In this research they used COQUALMO prediction model to predict the fault in a software and applied various clustering algorithms like k-means, agglomerative clustering, COBWEB, density based scan, expectation maximization and farthest first. Implementation was done using Weka tool. Finally they conclude that k-means algorithm works better when compared with other algorithms [1].

Anuradha Chug et al. used three supervised [classification] learning algorithms and three unsupervised [clustering] learning algorithms for predicting defects in software. NASA MDP datasets were run by using Weka tool. Several measures like recall and f-measure are used to evaluate the performance of both classification and clustering algorithms. By analyzing different classification algorithms Random Forest has the highest accuracy of MC1 dataset and also yields highest value in recall, f-measure and receiver operating characteristic [ROC] curve and it indicates minimum number of root mean square errors in all circumstances. In an unsupervised algorithm k-means has the lowest number of incorrect clustered instances and it takes minimum time for predicting faults [3].

Jaechang Name et al. applied Hetrogeneous Defect Prediction [HDP] to predict defects in with-in and across projects with different datasets. Metric selection, metrics matching and building a prediction model are the 3 methods used in this work. In this research they used various datasets from NASA, PROMISE, AEEEM, MORPH and SOFTLAB. Source and target datasets are used with different metric sets. For selecting metrics feature selection techniques such as gain ratio, chi-square, relief-F and significance attribute selection are applied to the source. To match source and target metrics various analyzers like Percentile based Matching (PAnalyzer), Kolmogorov – Smirnov test based matchiong (KSAnalyzer), Spearman's Correlation based Matching (SCOAnalyzer) are used. Cutoff threshold value is applied to all pair scores and poorly matched metrics are removed by comparison. Area Under the Receiver Operator Characteristic Curve [AUC] measure is used to compare the performance between different models. HDP is compared with 3 baselines – WPDP, CPDP-CM, CPDP-IFS by applying win/loss/tie evaluation. The experiments are repeated for 1000 times and Wilcoxon signed rank test ($P<0.05$) is applied for all AUC values and baselines. Performance is measured by counting the total number of win/loss/tie. When a cutoff threshold value gets increased in PAnalyzer and KSAnalyzer, the results (win) also gets increased. Logistic

Regression (LR) model works better when there is a linear relationship between a predictor and bug-prone [4].

Logan Perreault et al. applied classification algorithm such as naïve bayes, neural networks, support vector machine, linear regression, K-nearest neighbor to detect and predict defects. The authors used NASA and tera PROMISE datasets. To measure the performance they used accuracy and f1 measure with clearly well defined metrics such as McCabe Metrics and Halstead Metrics. 10-fold cross validation is used in which 90% of data are used for training and 10% of data are used for testing. ANOVA and tukey test was done for 5 dataset and 5 response variables. 0.05 is set as significance level for PC1, PC2, PC4 and PC5 dataset and 0.1 as PC3 dataset. Weka tool is used for implementation. Implementations of these 5 algorithms are available on Github repository. Finally the authors conclude that all datasets are similar and they are written in C or C++ and in future the work can be extended by selecting the datasets that are written in Java and instead of using weka tool for implementation some other tool can also be used [5].

Ebubeogu et al. employed predictor variables like defect density, defect velocity and defect introduction time which are derived from defect acceleration and used to predict the total number of defects in a software. MAChine – Learning – Inspired [MACLI] approach is used for predicting defects. The proposed framework for defect prediction has two phases. 1) Data pre- processing phase. 2) Data analysis phase [6].

Rayleigh distribution curve is a proposed modeling technique used to identify predictor variables and indicates the number of defects involved in developing SDLC. Simple linear regression model and multiple linear regression models are used to predict the number of defects in software. The authors conclude that defect velocity performed best in predicting the number of defects with the strongest correlation co-efficient of 0.98.

Yongli et al. applied data filters to datasets in order to increase the performance of CPDP. In this research the authors proposed Hierarchical Select-Based Filter [HSBF] strategy. HSBF is based on hierarchical data selection from software project level to software module level. Max value, min value, mean value and standard deviation are the four indicators which are merged together to represent the distributional characteristic of a given project. To correct the inconsistencies in software metrics between projects, cosine distance is applied. In this study, PROMISE datasets and Confusion matrix are used to evaluate the performance measure. Due to imbalanced dataset probability of detection [pd], probability of false alarm [pf] and AUC are also applied to measure the performance. Therefore the authors conclude from the experiments, Naïve Bayes [NB] algorithm performs better than Support Vector Machine. For smaller projects Target - Project Data Guided Filter [TGF] is used and for larger projects Hierarchical Select Based Filter [HSBF] is used for data selection from multi-source projects [7].

Xiao Yu et al. build a prediction model for Cross Company Defect Prediction [CCDP] by applying six imbalance learning methods such as under sampling techniques (random under sampling and near miss), over sampling techniques [SMOTE and ADASYN] and oversampling followed by under sampling [SMOTE Limks, TOMEK, SMOTE ENN] [8]. PROMISE datasets and classification algorithms such as NB, Random Forest [RF] and Linear Regression [LR] are applied. Probability of detection, probability of false alarm and g-measure are used to measure the performance. The authors conclude that NB performs better in predicting defects and it has a high pf value. Under sampling method works better with g-measure.

Shamsul Huda et al. [9] studied that developing a defect prediction model by using more number of metrics is a tedious process. So that a subset of metrics can be determined and selected. In this research two novel hybrid SDP models such as wrappers and filters are used for identifying the metrics. These two models combine the training of metric selection and fault prediction as a single process. In this research different datasets and classification algorithms such as Support Vector

Machine [SVM] and artificial neural network are used. Performance was measured by using AUC and MEWMA (Multivariate Exponentially Weighted Moving Average), implementation was done by using liblinear tool and mine tool.

Gopi Krishnan et al. applied regression models to build a defect classifier. In this work tera- PROMISE defect datasets and machine learning algorithms such as Linear/Logistic Regression, RF, K-Nearest Neighbour, SVM, CART and Neural Networks are used to build a prediction model. Two defect classifiers are developed namely discretized defect classifier and regression based defect classifier. In this work AUC is applied to evaluate the performance of a model and the authors conclude that there is a loss of information in discretized classifier. Regression-based classifier uses continuous defect counts as the target variable for determining a defect module.

Xinli Yang et al. proposed TLEL [Two Layer Ensemble Learning] to predict defects at change level. The advantages of ensemble methods are:

- Better performance can be achieved compared with single classifier.
- Combines bagging and stacking methods.

TLEL has two layers namely inner layer and outer layer. In an inner layer, decision tress and bagging are merged to develop a random forest model. In an outer layer, random under sampling is used to train various random forest models and stacking is used to train ensemble techniques. TLEL is compared with 3 baseline methods such as deeper, DNC and MKEL. Performance is measured by using cost effectiveness and F1-Score.

III. METHODOLOGY

Mostly three approaches are performed to evaluate prediction models.

1. With-in Project Defect Prediction

A prediction model can be constructed by collecting historical data from a software project and predicts faults in the same project are known as WPDP. WPDP performed best, if there is enough quantity of historical data available to train models.

Turhan, Burak, et al. [15] suggested that software defect prediction areas typically focus on developing defect prediction models with existing local data (i.e. within project defect prediction). To apply these models, a company should have a data warehouse, where project metrics and fault related information from past projects are stored.

Zimmermann et al. [11] notify that defect prediction performs better within projects as long as there is an adequate data to train models. That is, to construct defect predictors, we need access to historical data. If the data is absent, Cross Company Defect Prediction (CCDP) can be applied.

The drawbacks of with-in project defect prediction are:

- It is not constantly possible for all projects to collect such historical data
- Hence 100% accuracy cannot be achieved using WPDP.

On the other hand, historical data is often not presented for new projects and for many organizations. In this case, successful defect prediction is complicated to accomplish. To tackle this problem, cross project defect prediction strategy was applied.

2. Cross Project Defect Prediction [CPDP] for Similar Dataset

CPDP is used in a mode such that a project does not have sufficient historical data to train a model. So that, a prediction model is developed for one project and it has been applied for some other project or across project. i.e., transferring prediction models from one project to another project [10]. The drawbacks of applying CPDP is that it desires projects that have similar metric set, implication that the metric sets must be equal among projects. As an outcome, present techniques for CPDP are complicated to relate across projects with dissimilar dataset.

3. Cross Project Defect Prediction [CPDP] for Hetrogeneous Dataset

To deal with the inadequacy of using only similar dataset for CPDP, heterogeneous defect prediction [HDP] technique was proposed to predict defects across projects with imbalanced metric sets [4].

Defect Prediction Techniques

To improve the effectiveness and quality of software development and to predict defects in software, various data mining techniques can be applied to different Software Engineering areas. The broadly used SDP techniques are datamining techniques and machine learning techniques are depicted in Figure 1.

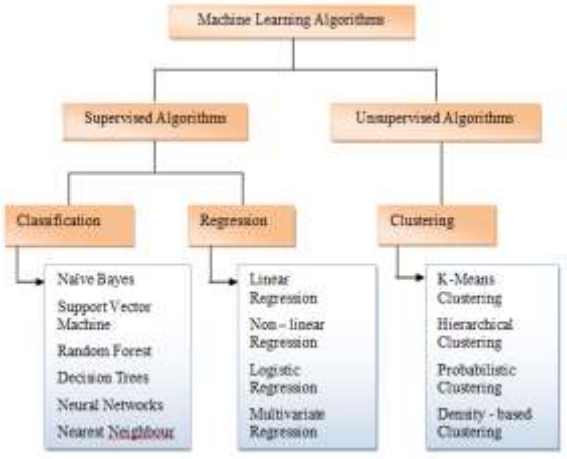

Figure 1: Machine learning algorithms

To predict a fault in software various data mining techniques are applied. In data mining, learning can be of two types:

- Supervised Learning
- UnSupervised Learning

Supervised Learning

Learning techniques are intended to determine whether software module has a higher fault hazards or not. In supervised learning data is extracted using the target class.

If machine learning task is trained for each input with consequent target, it is called supervised learning, which will be able to provide target for any new input after adequate training. Targets expressed in some classes are called classification problem.

If the target space is continuous, it is called regression problem. All classification and regression algorithms appear under supervised learning. Some of the supervised learning algorithms are:

- Decision tree classification algorithm
- Support vector machine (SVM)
- k-Nearest Neighbors
- Naive Bayes
- Random forest
- Neural networks
- Polynomial regression
- SVM for regression

Logan Perreault et al. [5] applied classification algorithm such as naive bayes, neural networks, support vector machine, linear regression, K-nearest neighbour to detect and predict defects.

Ebubeogu et al. employed [6] simple linear regression model and multiple linear regression model to predict the number of defects in a software.

Regression Techniques

A variety of regression techniques have been proposed in predicting amount of software defects [15]. A regression technique is a predictive modeling technique which examines the association among a dependent (target) and independent variable (s) (predictor). Commonly used regression techniques are:

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Lasso Regression
- Multivariate Regression

Unsupervised Learning

In an unsupervised learning, there is no previous information and everything is done dynamically. If the machine learning task is trained only with a set of inputs, it is called unsupervised learning [3], which will be able to find the structure or relationships between different inputs. Most important unsupervised learning is clustering,

which will create different cluster of inputs and will be able to place new input in an appropriate cluster. All clustering algorithms come under unsupervised learning algorithms.

- K – Means clustering
- Hierarchical clustering
- Make Density Based Clustering.

Software Metrics

Extensive investigation has also been carried out to predict the number of defects in a component by means of software metrics. Software metrics is a quantitative measure which is used to assess the progress of the software. Three parameters are used and measured as depicted in Figure 2.

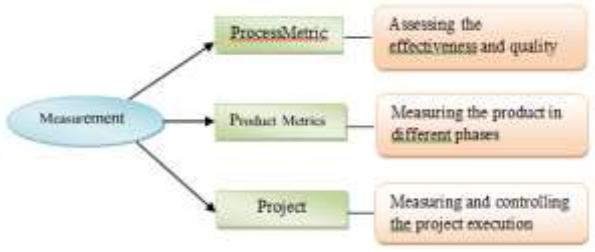

Figure 2: Various parameters of software metrics

Process metrics assess the efficacy and worth of software process, determine maturity of the process, effort required in the process, effectiveness of defect deduction during development, and so on. Product metrics is the measurement of work product created during different phases from requirements to deployment of a software development. Project metrics are the measures of software project and are used to monitor and control the project execution.

Objectives of Research

- Quantitatively measuring the size of the software
- Complexity level is assessed.
- Identifying the release date of the software
- Estimation is done on resources, cost and schedule.

Software Metrics are used for Defect Prediction

- LOC metric
- Cyclomatic Complexity (McCabe's Complexity)
- Halstead Metrics
- McCabe Essential Complexity(MEC) Metric
- The McCabe Module Design Complexity (MMDC) metric
- Object oriented metrics.

IV. SOFTWARE DEFECT DATASET

The fault prediction dataset is a group of models and metrics of software systems and their histories. The aim of such a dataset is to permit people to evaluate different fault prediction approaches and to evaluate whether a new technique is an enhancement over existing ones. PROMISE, AEEEM, ReLink, MORPH, NASA, and SOFTLAB [4] are the defect datasets which are publically available to the user.

Anuradha Chug et al. [3] used numerous NASA defect datasets for predicting defects using supervised and unsupervised learning algorithms.
Jaechang Nam et al. [4] applied various defect datasets includes NASA, PROMISE, AEEEM, SOFTLAB and MORPH for predicting defects by using machine learning algorithms..

Performance Measures

Performance measures are used to evaluate the accuracy of a prediction model. A prediction model can be constructed by using both classification and clustering algorithms [3]. Separate performance measures are available for both classification and clustering techniques.

Xiao Yu et al. [8] applied probability of detection (pd), probability of false alarm (pf) and g- measure as measure to evaluate the performance of a defect prediction model..

Gopi Krishnan et al. [13] used AUC to measure the performance of a developed defect prediction model.

Some of the classification evaluation measures are:

- Recall
- Precision
- F-measure
- G-measure
- AUC

- ROC
- Mean absolute error(MAE)
- Root mean square error(RMSE)
- Relative absolute error and accuracy(RAE)

Some of the clustering evaluation measures are:

- Time taken
- Cluster instance
- Number of iterations
- Incorrectly clustered instance
- Log likelihood

V. CONCLUSION

At present the growth of software based system are rising from the previous years due to its advantage. On the other hand, the quality of the system is essential prior it is delivered to end in order to improve the efficiency and quality of software development, software faults can be predicted at early phase of life cycle itself. To predict the software faults a variety of data mining techniques can be used. The key objective of this study was to assess the previous research works with respect to software defect which applies data mining techniques, datasets used, performance measures used, tools they used, and we classified it in to three such as based on classification, clustering and regression methods.

REFERENCES

1. Grishma, B. R., and C. Anjali. "Software root cause prediction using clustering techniques: A review." Communication Technologies (GCCT), 2015 Global Conference on. IEEE, 2015.
2. He, Peng, et al. "An empirical study on software defect prediction with a simplified metric set." Information and Software Technology 59 (2015): 170-190.
3. Chug, Anuradha, and Shafali Dhall. "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm." (2013): 5-01.
4. Nam, Jaechang, et al. "Heterogeneous defect prediction." IEEE Transactions on Software Engineering (2017).
5. Perreault, Logan, et al. "Using Classifiers for Software Defect Detection." 26th International Conference on Software Engineering and Data Engineering, SEDE. 2017.
6. Felix, Ebubeogu Amarachukwu, and Sai Peck Lee. "Integrated Approach to Software Defect Prediction." IEEE Access 5 (2017): 21524-21547.
7. Li, Yong, et al. "Evaluating Data Filter on Cross-Project Defect Prediction: Comparison and Improvements." IEEE Access 5 (2017): 25646-25656.
8. Yu, Xiao, et al. "Using Class Imbalance Learning for Cross-Company Defect Prediction." 29th International Conference on Software Engineering and Knowledge Engineering (SEKE 2017). KSI Research Inc. and Knowledge Systems Institute, 2017.
9. Huda, Shamsul, et al. "A Framework for Software Defect Prediction and Metric Selection." IEEE Access (2017).
10. Ni, Chao, et al. "A Cluster Based Feature Selection Method for Cross-Project Software Defect Prediction." Journal of Computer Science and Technology 32.6 (2017): 1090- 1107.
11. Zimmermann, Thomas, et al. "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process." Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2009.
12. Laradji, Issam H., Mohammad Alshayeb, and Lahouari Ghouti. "Software defect prediction using ensemble learning on selected features." Information and Software Technology 58 (2015): 388-402.
13. Rajbahadur, Gopi Krishnan, et al. "The impact of using regression models to build defect classifiers." Proceedings of the 14th International Conference on Mining Software Repositories. IEEE Press, 2017.
14. Yang, Xinli, et al. "TLEL: A two-layer ensemble learning approach for just-in-time defect prediction." Information and Software Technology 87 (2017): 206-220.
15. Turhan, Burak, et al. "On the relative value of cross-company and within-company data for

defect prediction." Empirical Software Engineering 14.5 (2009): 540-578.