JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Software Defect Prediction using Machine Learning Algorithms

Karedla Chayadevi
Dept. Of Computer Science Engineering,
Specialization in CNIS,
Andhra University College of Engineering, Visakhapatnam,
Andhra Pradesh, India

Prof. Ch.Satyanada Reddy
Dept. Of Computer Science Engineering,
Andhra University College of Engineering, Visakhapatnam,
Andhra Pradesh, India

Abstract—Software testing is a time-consuming and costly task, as it involves testing all software modules. To minimize the cost and effort of software testing, automatic defect detection can be used to identify the defective modules during the early stages. These aid software testers in detecting the modules that require intensive testing. Therefore, automatically predicting software defects has become a critical factor in software engineering. This paper explores the existing methods and techniques on software defect prediction (SDP) and lists the most popular datasets that are used as benchmarks in SDP. In addition, it discusses the approaches to overcome the class imbalance problem, which usually occurs in the benchmark datasets for SDP problems. This paper can be helpful for researchers in software engineering and other related areas by using machine learning algorithms. We use naïve bayes and random forest algorithms for software defect prediction.

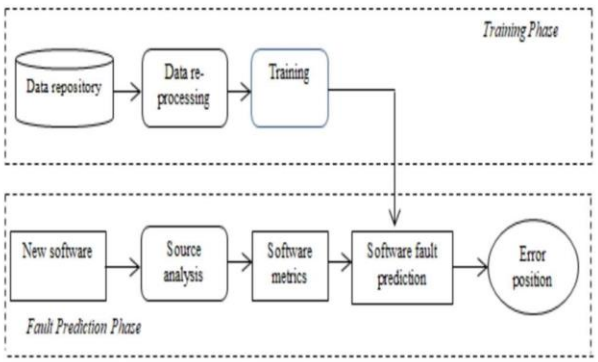
Keywords—Software Defect Prediction, Machine Learning, Random Forest, Naïve Bayes, NASA Promise Repository

I. INTRODUCTION

Usage of devices in our day-to-day lives has increased our dependency on various software systems. Any disruption in the working of software requiring high dependency can have serious consequences. To ensure error-free working of the systems, software must undergo a thorough testing process.

In the process of developing software, testing process of the software plays a crucial role. However, experienced a developer is, there is always a chance that there might be unforeseen defects. Software can also fail if defects get introduced during maintenance phases. This makes software testing a very significant stage in the software development life cycle. Due to this reason, it is highly desirable to use the testing resources efficiently so that quality of the software is improved. Software Defect Prediction provides a mechanism for effective and efficient usage of testing resources by early prediction of modules in software as defect prone or non-defect prone. It's an essential and continuous activity to improve the software quality. The primary purpose of software defect prediction is to deliver an error free product and assure the software quality. It is helpful to identify the defect or error at the earliest and contribute to the quality assurance mechanism to make a reliable software application. In recent research works, machine learning is being used extensively to create defect prediction models. Various techniques have been recommended to solve the task of software defect prediction. These techniques made predictions on the grounds of the historical defect data, the software metrics as well as the algorithm using which predictions are to be done. Classification, clustering and regression are the frequently used techniques for Machine learning algorithms such as random forest and naïve bayes.

Therefore, the aim of this work is to design machine learning models that provide more accurate results in detecting if a software module is defect prone or not and help in finding the undiscovered defects. This can be achieved by extensive analysis of the software metrics and the features of the dataset. Useful features need to be filtered from the feature set to have clean data that can be analyzed properly. This research work focuses on combining feature extraction and feature selection method with an aim to get more accurate results. The prediction of software defects can minimize the effort, time, and cost of software development.



II. LITERATURE SURVEY

A survey for detecting and predicting of diabetes using machine learning techniques. This paper focuses on machine learning promising the improving accuracy of perception and diagnosis of the diseases. The various machine learning techniques that are used to classify the
datasets include supervised, unsupervised, reinforcement, semi-supervised, and deep learning, evolutionary learning algorithms. It also shows the comparison of the two methods namely, Naïve Bayes and Random Forest (RF). The Bayesian Network applies the Naïve Bayes theorem which firmly assumes that the presence of any attribute in a class is not related to the presence of any other attribute, making it much more advantageous, efficient and independent. The paper starts with the introduction section, which contains four subsections detailing the significance, evolution timeline, motivation for writing this paper, and background of defect identification and the prediction. The background section provides common defect identification and prediction practices using manual testing, automation testing, and prediction approaches. The prior research section discusses a few significant studies on Software Defect Prediction and states the research goals and the study's contribution. The terminology section briefly describes commonly used terms in Software Defect Prediction. The research methodology section describes the SLR process. We followed the Kitchenham (Kitchenham et al., 2009) guidelines.

The research methodology section contains the keywords used for searching the research studies in various databases. Selection criteria talk about how the papers are selected for this SLR. Inclusion and Exclusion criteria discuss the criteria used to select and filter the papers. The selection of the paper is made based on the various parameters and the score count of the paper. The literature outcome section describes the analysis of the previous studies against the formulated research questions. The discussion section summarizes the literature outcome on formulated research questions. The limitation of the study section discusses possible limitations present in this SLR. The conclusion section concludes the findings as per the analysis. Finally, the future work and opportunity section discusses the proposed research work based on the limitation found in the earlier research.

The existing literature is lacking on systematic literature reviews for Software Defect Predictions. Earlier research is mainly focused on defect classification using publicly available datasets. Prediction outcome is limited, and there are no actionable items for the software development team. There is a lack of a survey that focuses on the legacy versus modern approaches for identifying software defects. Earlier surveys did not cover the existing tools. Before using the data to train the classifiers, the adequate focus was not given to the data validation techniques. Hence, a comprehensive survey needs to focus on datasets, data validation methods, defect detection and prediction approaches, tools, and recommendations for further research.

This Systematic Literature Review focuses on datasets, methods for data validation, defect detection and prediction approaches, tools, and recommendations for future researchers. According to a bibliometric review (Pachouly et al., 2020), there is a lot of interest in the field of Software Defect Prediction among researchers a worldwide.

III. SOFTWARE DEFECT PREDICTION USING MACHINE LEARNING Algorithms

Machine learning is a sub-field of artificial intelligence that learns the patterns in the data to develop performance prediction and classification tasks. Statistical and machine-learning techniques have been used by many researchers to predict the defective prone software modules. The machine-learning methods have been proven to be effective at identifying the defective modules.

As software defect prediction project can be most interesting and most organizations want to predict the number of defect-prone modules in software systems before they are deployed. This can be managed by numerous statistical methods and artificial intelligence techniques have been employed in order to predict defects that a software system will reveal in operation or during testing. This is a temporary and unique endeavor designed to produce a product, service or result with a defined begging and end. Time, cost and quality are the building blocks of every project.

Requirement specification is the application of processes, methods, skills, knowledge and experience to achieve specific project objectives according to the project acceptance criteria within agreed parameters.

Introduction to NAÏVE BAYES

The Bayesian Network plays an important part of machine learning in classification or prediction of defects. The most commonly used type of Bayesian Network for classification is the Naïve Bayesian's, which has the highest accuracy value of up to 99.51% respectively. The Bayesian Network applies the Naïve Bayes theorem which firmly assumes that the presence of any particular attribute in a class is not related to the presence of any other attribute, making it much more advantageous, efficient and independent.

The Bayesian Network is one of the most used techniques in the classification of software defects, which has an accuracy in the range of 71% to 99.51%. The Naïve Bayesian is based on the conditional probability (given a set of features, the probability of a certain results occurrence).

Since this method is mainly used in text classification, this algorithm is found to have a huge success rate when compared to another algorithm. This success rate makes it a very efficient algorithm for the prediction of diabetes and gives good percentages of accuracy while performing analysis. It can be used for the purpose of spam filtering where all the spam emails in our inbox are stored in the spam folder. This helps in separating the important
messages from messages that are sent for the intent of phishing, virus, etc.

Naïve Bayes is a very fast algorithm, so it can be used for the purpose of real time predictions. When it comes to diabetes prediction, the dataset is used to analyse and also predict if a person has diabetes or not.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \text{ Where } X = (X_1, X_2, X_3, X_4, \ldots\ldots\ldots\ldots X_n)$$

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

$$P(y|x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Fig.: Naïve bayes formula

INTRODUCTION TO RANDOM FOREST

The Random Forests algorithm is a powerful classification algorithm that can classify large amounts of data with high accuracy. Random Forest is a group learning method (it's a form of the nearest neighbour predictor) for classification and regression that build a number of decision trees at training time and display the class that is the mode of the classes output by individual trees.

They are a combination of tree predictors where each tree depends on the values of a random vector sampled independently with the same distribution for all trees in the forest. Random Forests solves this problem of high variance and high bias by finding a natural balance between the two extremes.

They also have a mechanism to estimate the error rates (Out of the Bag error). Many machine learning models, like linear and logistic regression are easily impacted by the outliers in the training data. Outliers are changes in the system behaviour and can also be caused by human error, instrument error. There are chances for a given sample to be contaminated. These outliers or extreme values do not impact the model performance/accuracy. RF Algorithm overcomes and solves this problem. The decision treasures are generated by a selection indicator for each attribute, such as information gain, gain ratio and Gini index. The independent sample size depends on each tree. Each tree voting and the most popular class is considered the optimal result in a classification problem.

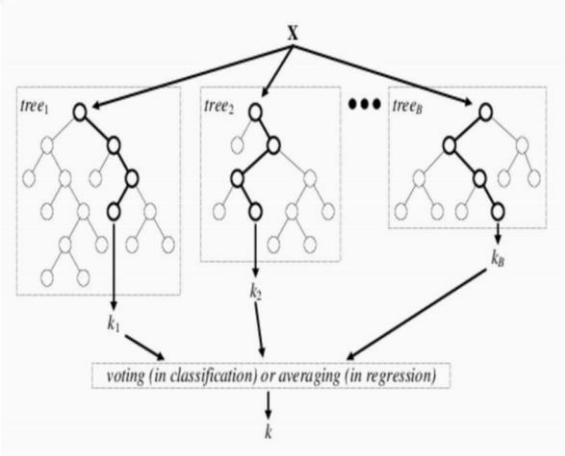


Fig: Random Forest Architecture

IV. DATASET

For our project software defect prediction, we taken our datasets from the NASA repository, The NASA PROMISE Repository is a public dataset repository that provides a collection of software engineering datasets for research purposes. The repository was established in 2006 as part of the NASA Software Engineering Laboratory's research activities and contains datasets from various domains of software engineering. The datasets are designed to support the development and evaluation of software engineering techniques, including software defect prediction, software effort estimation, software quality assurance, and software maintenance.

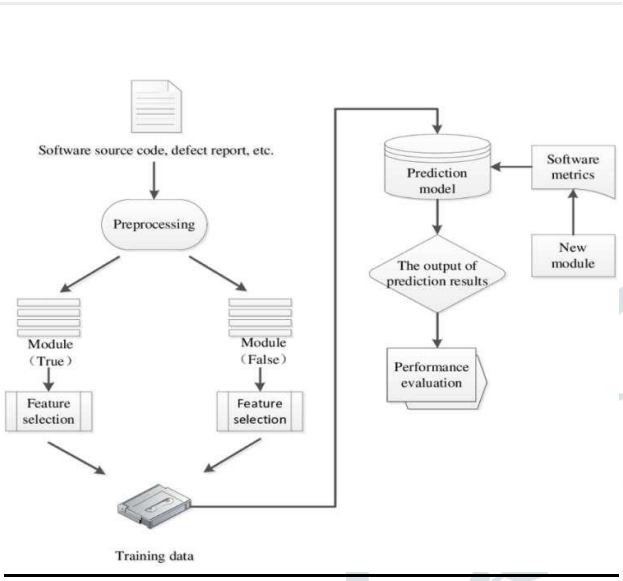
The PROMISE repository currently contains over 50 datasets from various software engineering domains. The datasets are collected from publicly available sources, such as open-source software repositories, bug tracking systems, and software development projects. Each dataset includes a set of features, such as lines of code, number of developers, and complexity metrics, and a target variable, such as the number of defects, the effort required, or the quality of the software.

The details of projects from the NASA/PROMISE and other relevant datasets repository used for the experimentation (http: promise.site.uotawa.ca, Repository, datasets-page.html.) We have experimented on the 11 state-of-art projects, namely PC1, PC2, PC3, KC1, KC2, CM1, ANT, CAMEL, IVY, LOG4J and TOMCAT. The features selected from PC1, PC2, PC3, KC1, KC2, CM1, ANT, CAMEL, IVY, LOG4J and TOMCAT data sets are
described. The features listed above are the software metrics computed for analyzing the quality of software systems based on state-of-art quality metrics.
In our project we are using KC1, KC2, PC1, JM1, and CM1 and those are all software defect prediction datasets.

SPLITTING THE DATASET

Splitting the data into training and test data, is one of the most crucial steps in the analysis. The split of the training data is more than the training data. The training data undergoes through learning. This data which trained is later generalized on the other data, based on which the prediction is made. The dataset in our case, is split into multiple variants and prediction is performed accordingly. The dataset has multiple column that are medical predictors and one target column, that of the diabetes outcome. The medical predictors are given as inputs to a variable and the target variable is input to another variable.

Using the inbuilt function, train_test_split, the dataset is split into arrays and is mapped to training and test subsets. In our case, we are performing splits of 80/20,70/30,75/25,60/40 and the accuracy of each is recorded. It was noticed that the dataset contained values that were null, hence in order to streamline the analysis and the prediction, the null values were filled with the mean values of the respective columns.



V.TRAINING AND TESTING

Classifiers are grouped into two categories, Statistical Approach and rf and nb. Data is selected according to classifiers need, so as to achieve balance between techniques. Each dataset is partitioned randomly into two sets: a) Training set; b) Test set. Partitioning is done by means of split sample setup, using 2/3rd and1/3rd of data values, respectively. This is done to protect the class allotment and for performance estimation. It is assumed that spilt-sample setup is the key to calculate accuracy in fault prediction models.

Metrics	Description
LOC	Sum of line in the module
iv(g)	Design complexity of each module
ev(g)	Essential complexity of each module
N	Sum of operators and operands existing in the module
v(g)	Cyclomatic complexity of each module
D	Difficulties in each module
B	Effort approximation
L	Program size for each module
V	Volume of each module
I	Intelligence content
E	Error approximation
Locomment	Line of comments in each module
Loblank	Sum of blank lines in each module
uniq_op	Sum of unique operators
uniq_opnd	Sum of unique operand
T	Time determinist
Branchcount	Sum of branch in the software module
total_op	Sum of operators
total_opnd	Sum of operators
Locodeandcomment	Sum of line of code and comments
Defects	Details on whether there is existence of defect or not

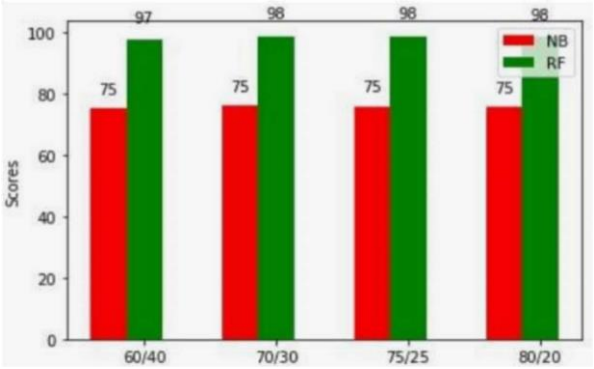
Data preprocessing:

Data preprocessing is a crucial step in software defect prediction, which involves transforming raw data into a format suitable for analysis. The aim of data preprocessing is to improve the quality of data and to remove any inconsistencies, errors, or redundancies in the data that might affect the accuracy of the predictions. The preprocessing phase is an important part of the overall data mining process and includes several techniques to extract meaningful insights from the data.

To predict values using the training data, we use the predict function. A class prediction is given the finalized model and one or more data instances, predict the class for the data instances. We do not know the outcome classes for the new data. That is why we need the model in the first place. We can predict the class for new data instances using our finalized classification model in scikit-learn using the predict () function.

Table: Prediction Using Naïve Bayes

Train	Test	Train_Result (% value)	Test_Result (% value)
60	40	75.22%	77.27%
70	30	75.98	74.89
75	25	75.87	74.48
80	20	75.57	77.27



the best prediction result is giving by the 60/40 split while performing Random Forest.

VI. TABULATED RESULTS

After performing the Random Forest and Naive Bayes algorithms, we are generating the following results for the different splits of training and testing data:

In the above table, we can see that for the four different splits, we get results that are close to 75% in the training set and 74-77% in the test results.

This depicts that the training set has been trained up to 75% accuracy which means that the data that has been trained has been used to predict the test results which have a 75% average accuracy in the analysing of the dataset. For each split, the percentage of test results depicts that 74-77% of the dataset prediction is accurate and rest of the 25% approx. cannot be predicted due to various other reasons.

Table: Prediction Using Random Forest

Train	Test	Train_Result (% value)	Test_Result (% value)
60	40	97.61	77.27 (best)
70	30	98.70	73.16
75	25	98.61	72.92
80	20	98.37	74.68

In the above table, we can see that for the four different splits, we get results that are close to 98% in the training set and 72-77% in the test results. This depicts that the training set has been trained up to 98% accuracy which means that the data that has been trained has been used to predict the test results which have a 75% average accuracy in the analysing of the dataset. For each split, the percentage of test results depicts that 72-77% of the dataset prediction is accurate and rest of the 25% approx. cannot be predicted due to various other reasons.

While analysing both the tables, we can understand that the Random Forest algorithm has a better training set result which in turn gives a better accuracy of the prediction and analysis. The dataset is trained to the maximum accuracy where all variables are taken into aspect without excluding missing data as Random Forest algorithm will make sure that there is no missing data in large datasets. Naïve Bayes algorithm tends to ignore missing data which does not provide accurate results while performing analysis. From the tables, we can find out that

COMPARISON GRAPHS

The training result of Naïve Bayes is very low compared to that of Random Forest as there are errors that occur in the Naïve Bayes algorithm while performing training. Sometimes, it cannot detect missing data so there are fluctuations and errors in the accuracy of the result,

Fig: Comparison of Training results for various splits The above graph depicts the comparison graph for the training results for both Naïve Bayes and Random Forest for various splits. We can understand that the Random Forest training results are more accurate when compared to that of Naïve Bayes as it gives a 98% accuracy when it comes to training the dataset.

The training result of Naïve Bayes is very low compared to that of Random Forest as there are errors that occur in the Naïve Bayes algorithm while performing training. Sometimes, it cannot detect missing data so there are fluctuations and errors in the accuracy of the result, but in the case of Random Forest, it gives the proper accuracy even when it comes to large datasets like NASA dataset.

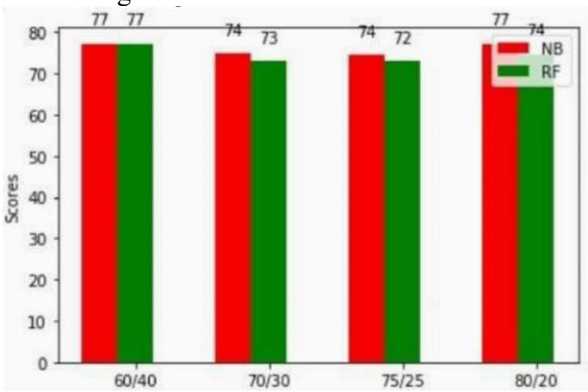


Fig.: Comparison of Test results for various splits

The above graph depicts the comparison graph for the testing results for both Naïve Bayes and Random Forest for various

splits. We can understand that the Random Forest and Naïve Bayes test results are almost the same and they differ by 2-3%. Even though the Naïve Bayes testing results are greater compared to the Random Forest results, the training result for Naïve Bayes was lesser than that of Random Forest, so the accuracy of the results when compared, is greater for Random Forest since the training data was much more accurate when compared to Naïve Bayes.

After analysing the results, we can come to the conclusion that the Random Forest algorithm is a more efficient method to analyse the dataset using means of splitting it into training and testing sets. It serves as a more accurate method of prediction of software defects.

VII. CONCLUSION

Software metrics are often used for evaluating and improving software quality. We studied the using of object-oriented metrics in applying different machine learning techniques for predicting fault. Our experimental study was conducted on seven popular machine learning techniques for predicting defects by using the PROMISE datasets at both method-level and class-level. The obtained results conclude that Naïvebayes has the highest performance for class-level datasets and Random Forest performs other techniques for method-level datasets.

This paper reviewed the existing datasets and methods that are used in SDP. This is because software quality assurance is a critical but expensive portion of the lifecycle of software. Identifying software defects prior to the testing stage can reduce both maintenance costs and time. Predicting software defects helps with identifying the defect-prone modules using software metrics. Future works will focus on proposing a software defect prediction scheme based on the machine learning algorithm to study the impact of feature selection with oversampling technique. The ML algorithms are combination of the feature-selection technique and machine learning classifiers. It yields the most effective and reliable results. The two major issues of software defect prediction are the data imbalance and reduced dimensionality. The oversampling and feature-selection methods aim to resolve these two issues. In the future work, we will study the classification techniques in order to deal with the imbalance issue of datasets for defect prediction.

REFERENCES

1. Hauer F, Pretschner A, Schmitt M, Gro¨tsch M (2017) Industrial evaluation of search-based test generation techniques for control systems. In: The 28th international symposium on software reliability
engineering (ISSRE)

2. Yalc¸ıner B, O¨ zdes¸ M (2019) Software defect estimation using machine learning algorithms. In: 4th international conference on computer science and engineering (UBMK), Samsun, Turkey, pp 487–491. https://doi.org/10.1109/UBMK.2019.8907149

3. Shirabad JS, Menzies TJ (2005) The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Ottawa

4. Shenvi AA (2009) Defect prevention with orthogonal defect classification. In: Proceeding ISEC ’09 proceedings of the 2nd India software engineering conference

5. Caglayan B, Tosun A et al (2010) Usage of multiple prediction models based on defect categories. In: Proceeding PROMISE ’10 proceedings of the 6th international conference on predictive models in software engineering 6. Wang S, Yao X (2013) Using class imbalance learning for software defect prediction.
IEEE Trans Reliab 62(2):434–443

7. Bennin KE, Keung J, Monden A, Phannachitta P, Mensah S (2017) The significant effects of data sampling on software defect prioritization and classification. In: Proceedings of the 11th ACM/IEEE international symposium on empirical software engineering and measurement, IEEE Press, pp 364–373

8. Malhotra R (2015) A systematic review of machine learning techniques for software defect prediction. Appl
Soft Comput J 27:504–518

9. Reddivari S, Raman J (2019) Software quality prediction: an investigation based on machine learning. In: IEEE 20th International conference on information reuse and integration for data science (IRI), Los Angeles, CA, USA, pp 115–122. https://doi.
org/10.1109/IRI.2019.00030

10. Yang X, Lo D, Xia X, Zhang Y, Sun J (2015) Deep learning for just-in-time defect prediction. In: IEEE international conference on software quality, reliability and security, Vancouver, BC, pp 17–26. https://doi.or