International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 7 July 2023, pp: 461-467 www.ijaem.net ISSN: 2395-5252

Software Defect Estimation Using Machine Learning Algorithms

Pappala Sasi, Saragadam Sridhar

Department of Master of Computer Science ,Miracle Educational Society Group of Institutions, Vizianagram–535216 (AP) India

Department of Master of Computer Science, Miracle Educational Society Group of Institutions, Vizianagram–535216 (AP) India

Date of Submission: 10-07-2023 Date of Acceptance: 20-07-2023

ABSTRACT

Software Engineering is a branch of computer science that enables tight communication between system software and training it as per the requirement of the user. We have selected seven distinct algorithms from machine learning techniques and are going to test them using the data sets acquired for NASA public promise repositories. The results of our project enable the users of this software to bag up the defects are selecting the most efficient of given algorithms in doing their further respective tasks, resulting in effective results.In this work we hvae used SVM and RF algorithms

Keywords: Support Vector Machines(SVM), Random Forest Algorithm(RFA)

I. INTRODUCTION

Developing a software system is an arduous process which contains planning, analysis, design, implementation, testing, integration and maintenance. A software engineer is expected to develop a software system on time and within limited the budget which are determined during the planning phase. During the development process, there can be some defects such as improper design, poor functional logic, improper data handling, wrong coding, etc. and these defects may cause errors which lead to rework, increases in development and maintenance costs decrease in customer satisfaction. A defect management approach should be applied in order to improve software quality by tracking of these defects. In this approach, defects are categorized depending on the severity and corrective and preventive actions are taken as per the severity defined. The selected machine learning algorithms for comparison are used for supervised learning to solve classification problems.

They are two tree-structured classifier techniques: (i) Bagging and (ii) Random Forests (RF); two neural networks techniques: (i) Multilayer Perceptron (MLP) and (ii) Radial Basis Function (RBF); two Bayesian classifier techniques: (i) Naive Bayes and (ii) Multinomial Naive Bayes; and one discriminative classifier Support Vector Machine (SVM).

In this paper author is evaluating performance of various machine learning algorithms such as SVM, Bagging, Naïve Bayes, Multinomial Naïve Bayes, RBF, Random Forest and Multilayer Perceptron Algorithms to detect bugs or defects from Software Components. Defects will occur in software components due to poor coding which may increase software development and maintenance cost and this problem leads to dis-satisfaction from customers.To detect defects from software components various techniques were developed but right now machine learning algorithms are gaining lots of popularity due to its better performance.

So, in this paper also author is using machine learning algorithms to detect defects from software modules. In this paper author is using dataset from NASA Software components and the name of those datasets are CM1 and KC1.

Support vector machine (SVM) is a supervised machine learning method capable of both classification and regression. It is one of the most effective and simple methods used in classification. For classification, it is possible to separate two groups by drawing decision boundaries between two classes of data points in a hyperplane. The main objective of this algorithm is to find optimal hyperplane.

II. LITERATURE SURVEY

[1] Victor R Basili, Lionel C. Briand, and Walcelio L Melo . ´ A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering, 22(10):751–761, 1996.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 7 July 2023, pp: 461-467 www.ijaem.net ISSN: 2395-5252

[2] Evren Ceylan, F Onur Kutlubay, and Ayse B Bener. Software defect identification using machine learning techniques. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EU-ROMICRO'06), pages 240–247. IEEE, 2006.
[3] Karim O Elish and Mahmoud O Elish. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5):649 660, 2008.
[4] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.
[5] Victor R Basili, Lionel C. Briand, and Walcelio L Melo. ´ A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering, 22(10):751–761, 1996.
[6] Evren Ceylan, F Onur Kutlubay, and Ayse B Bener. Software defect identification using machine learning techniques. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pages 240–247. IEEE, 2006.
[3] Karim O Elish and Mahmoud O Elish. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5):649– 660, 2008.
[7] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.
[8] Lan Guo, Yan Ma, Bojan Cukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In 15th International Symposium on Software Reliability Engineering, pages 417–428. IEEE, 2004.
[9] Taghi M Khoshgoftaar, Edward B Allen, and Jianyu Deng. Using regression trees to classify fault-prone software modules. IEEE Transactions on reliability, 51(4):455–462, 2002.
[10] Taghi M Khoshgoftaar, Edward B Allen, John P Hudepohl, and Stephen J Aud. Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Transactions on Neural Networks, 8(4):902–909, 1997

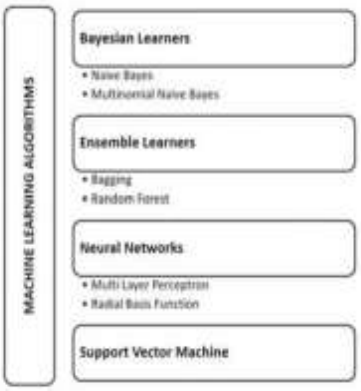
Proposed System :-

The proposed a model which uses three machine learning algorithms that are Decision Tree, Multilayer Perceptron and Radial Basis Functions in order to identify the impact of this model to predict defects on different software metric datasets obtained from the real*life projects of three big-size software companies in Turkey. The results have shown that all of the machine learning algorithms had similar results which have enabled to predict potentially defective software and take actions to correct them. have proposed a model to solve the class imbalance problem which causes a reduction in the performance of defect prediction. The Gaussian function has been used as kernel function for both the Asymmetric Kernel Partial Least Squares Classifier (AKPLSC) and Asymmetric Kernel Principal Component Analysis Classifier (AKPCAC) and NASA and SOFTLAB datasets have been used for experiments. The results have shown that the AKPLSC had better impact on retrieving the loss caused by class imbalance and the AKPCAC had better performance to predict defect on imbalanced datasets. There is also a systematic review study conducted by Malhotra to review the machine learning algorithms for software fault prediction.

Advantages :-

1. Performance measures are used to evaluate the accuracy of a prediction model.
2. Applied various defect datasets includes NASA, PROMISE, AEEEM, SOFTLAB and MORPH for predicting defects by using machine learning algorithms.
3. used AUC to measure the performance of a developed defect prediction model.

Architecture



SVM Algorithm: Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset. SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyper plane which separates the data into classes. In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 7 July 2023, pp: 461-467 www.ijaem.net ISSN: 2395-5252

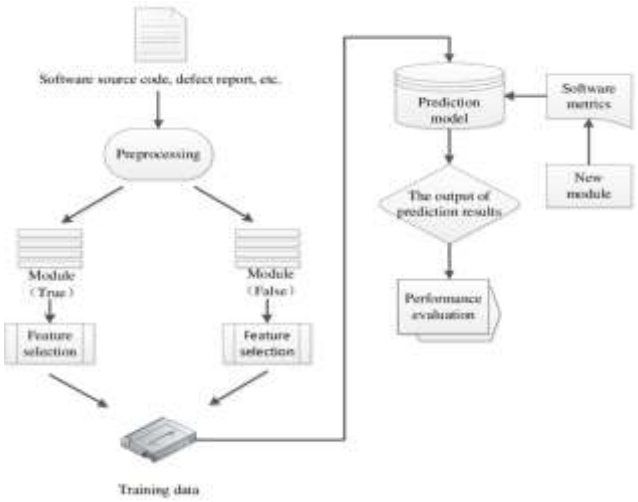
learning algorithms. In particular, it is commonly used in support vector machine classification. As a simple example, for a classification task with only two features (like the image above), you can think of a hyper plane as a line that linearly separates and classifies a set of data.

Intuitively, the further from the hyper plane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyper plane as possible, while still being on the correct side of it.

So when new testing data is added, whatever side of the hyper plane it lands will decide the class that we assign to it.

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.



Data Preprocessing

Clean and preprocess the collected data to ensure its quality and consistency. This may involve removing duplicates, handling missing values, normalizing data, and transforming it into a suitable format for machine learning algorithms.

Feature selection

Choose an appropriate machine learning algorithm for defect estimation, considering factors such as the nature of the data, the size of the dataset, and the goals of the project. Common algorithms used in defect estimation include decision trees, random forests, support vector machines (SVM), and neural networks.

Training Data:

Split the preprocessed data into training and validation sets. Use the training set to train the selecte machine learning model on the historical data, adjusting the model's parameters to optimize its performance.

Optimization

Fine-tune the model by experimenting with different techniques, such as hyperparameter tuning, feature selection, or ensemble methods, to improve its performance.

Performance Evaluation:

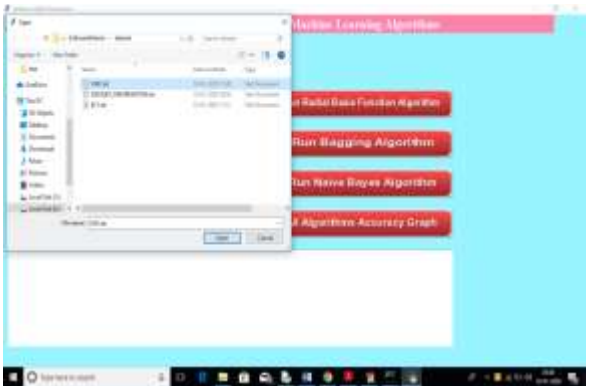
Evaluate the trained model's performance on the validation set using suitable metrics such as accuracy, precision, recall, and F1 score. This step helps assess the model's effectiveness in predicting software defects.

I am also using same datasets to evaluate performance of above mention algorithms.

To run this project double click on ‘run.bat’ file to get below screen

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 7 July 2023, pp: 461-467 www.ijaem.net ISSN: 2395-5252

In above screen click on 'Upload Nasa Software Dataset' button to upload dataset



In above screen uploading 'CM1.txt' dataset and information of this dataset you can read from internet of 'DATASET_INFORMATION' file from above screen.

After uploading dataset will get below screen

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 7 July 2023, pp: 461-467 www.ijaem.net ISSN: 2395-5252

In above screen we can see total dataset size and training size records and testing size records application obtained from dataset to build train model. Now click on 'Run Multilayer Perceptron Algorithm' button to generate model and to get its accuracy

In above screen we can see multilayer perceptron fmeasure, recall and accuracy values and scroll down in text area to see all details.

In above screen we can see multilayer perceptron accuracy is 93%. Similarly you click on all other algorithms button to see their accuracies and then click on 'All Algorithms Accuracy Graph' button to see all algorithms accuracy in graph to understand which algorithm is giving high accuracy.

In above graph x-axis represents algorithm name and y-axis represents accuracy of those algorithms. In all algorithms we can see MLP, Bagging is giving better accuracy

III. CONCLUSION

In this experimental study, seven machine learning algorithms are used to predict defectiveness of software systems before they are released to the real environment and/or delivered to

the customers and the best category which has the most capability to predict the software defects are tried to find while comparing them based on software quality metrics which are accuracy, precision, recall and F-measure. We carry out this experimental study with four NASA datasets which are PC1, CM1, KC1 and KC2.

These datasets are obtained from public PROMISE repository. The results of this experimental study indicate that tree-structured classifiers in other words ensemble learners which are Random Forests and Bagging have better defect prediction performance compared to its counterparts. Especially, the capability of Bagging in predicting software defectiveness is better. When applied to all datasets, the overall accuracy, precision, recall and FMeasure of Bagging is within 83,7-94,1%, 81,3-93,1%, 83,7- 94,1% and 82,4-92,8% respectively.For PC1 dataset, Bagging outperforms all other machine learning techniques in all quality metric.

However, Naive Bayes outperforms Bagging in precision and F-Measure while Bagging outperforms it in accuracy and recall for CM1 dataset. Random Forests outperforms all machine learning techniques in all quality metrics for KC1 dataset. Finally, for KC2 dataset, MLP outperforms all machine learning techniques in all quality metrics for KC2 dataset. It is deductive from obtained results that tree-structured classifiers are more suitable for software defect prediction. Moreover, it is recommended to software companies to utilize tree-structured classifiers for software defect prediction due to its performance. Utilizing these techniques enables them to save software testing and maintenance costs by identifying defects in the early phase of project life cycle and taking corrective and preventive actions before they becomes failures

The results of this experimental study indicate that tree-structured classifiers in other words ensemble learners which are Random Forests and Bagging have better defect prediction performance compared to its counterparts. Especially, the capability of Bagging in predicting software defectiveness is better. When applied to all datasets, the overall accuracy, precision, recall and FMeasure of Bagging is within 83,7-94,1%, 81,3-93,1%, 83,7- 94,1% and 82,4-92,8% respectively.For PC1 dataset, Bagging outperforms all other machine learning techniques in all quality metric.

However, Naive Bayes outperforms Bagging in precision and F-Measure while Bagging outperforms it in accuracy and recall for CM1 dataset. Random Forests outperforms all machine learning techniques in all quality metrics for KC1 dataset. Finally, for KC2 dataset, MLP outperforms all machine learning techniques in all quality metrics for KC2 dataset. It is deductive from obtained results that tree-structured classifiers are more suitable for software defect prediction. Moreover, it is recommended to software companies to utilize tree-structured classifiers for software defect prediction due to its performance.

REFERENCES

[1]. Victor R Basili, Lionel C. Briand, and Walcelio L Melo. ´ A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering, 22(10):751–761, 1996. \

[2]. Evren Ceylan, F Onur Kutlubay, and Ayse B Bener. Software defect identification using machine learning techniques. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pages 240–247. IEEE, 2006.

[3]. Karim O Elish and Mahmoud O Elish. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5):649– 660, 2008.

[4]. Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.

[5]. Lan Guo, Yan Ma, Bojan Cukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In 15th International Symposium on Software Reliability Engineering, pages 417–428. IEEE, 2004.

[6]. Taghi M Khoshgoftaar, Edward B Allen, and Jianyu Deng. Using regression trees to classify fault-prone software modules. IEEE Transactions on reliability, 51(4):455–462, 2002.

[7]. Taghi M Khoshgoftaar, Edward B Allen, John P Hudepohl, and Stephen J Aud. Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Transactions on Neural Networks, 8(4):902–909, 1997.

[8]. Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. Dealing with noise in defect prediction. In 2011 33rd International Conference on Software Engineering (ICSE), pages 481–490. IEEE, 2011.

[9]. Yan Ma, Lan Guo, and Bojan Cukic. A statistical framework for the prediction of fault-proneness. In Advances in Machine Learning Applications in Software Engineering, pages 237–263. IGI Global, 2007.

[10]. Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27:504–518, 2015

[11]. Jinsheng Ren, Ke Qin, Ying Ma, and Guangchun Luo. On software defect prediction using machine learning. Journal of Applied Mathematics, 2014, 2014.

[12]. J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[13]. Shuo Wang and Xin Yao. Using class imbalance learning for software defect prediction. IEEE Transactions on Reliability, 62(2):434–443, 2013. [14] Robert Andrew Weaver. The safety of software: Constructing and assuring arguments. University of York, Department of Computer Science, 2003.