International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Predicting Software Defects using Artificial Intelligence Technique

Manish Kumar Maurya, Dr. Rohitashwa Pandey
Department of Computer Science and engineering,
Bansal Institute of Technology, Lucknow, India,
manish22maurya@gmail.com

Abstract: Many researchers have applied various data mining, machine learning, deep learning methods to predict the software reliability. Researchers have often used the software failure data, which is made availabe by Musa (1979). The data set is of SPSS 14.0 availabe on http:// www.spss.com. The data set contains the cumulative failure time and inter failure time. We will use the machine learning models like ensemble learning with both boosting and bagging techniques, decision tree models, KNN, Linear regression models. All the models will be trained on the SPSS 14.0 dataset and then the multdimensional result analysis is performed to obtained the best model to predict the software reliabity. Now a days machine learning is been deployed in each and every field like medical, hospitality, automotive, customer behaviour, education, aerospace etc. to achive the best results.

Keywords: Attribute selection,Defect Prediction, Software Quality, Defect Detection.

1. Introduction:

Business applications which are critical in nature require reliable software, but developing such software's is a key challenge which our software industry faces today. With the increasing complexity of the software these days, achieving software reliability is hard to achieve. The primary goal of software reliability modeling is to find out the probability of a system failing in given time interval or the expected time span between successive failures

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (ANSI definition). Software reliability modeling has gained a lot of importance in the recent years. Criticality of software in many of the present day applications has led to a tremendous increase in the amount of work being carried out in this area. The use of intelligent neural network and hybrid techniques in place of the traditional statistical techniques have shown a remarkable improvement in the prediction of software reliability in the recent years. Among the intelligent and the statistical techniques it is not easy to identify the best one since their performance varies with the change in data. SRGM has been used for predicting and estimating number of errors remaining in the software

ISO 9126 defines software quality as "the totality of features and characteristics of software product that bears on its ability to satisfy stated or implied needs"
While ISO 25000 takes the following approach to quality "Capability of software products to satisfy stated and implied needs when used under specified conditions"

Software quality:- is the degree of conformance to explicit or implicit requirements and expectations.

- *Explicit*: clearly defined and documented
- *Implicit*: not clearly defined and documented but indirectly suggested
- *Requirements*: business/product/software requirements.
- *Expectations*: mainly end-user expectations.

2. Related Work:

In [1], Ai-jamimi and Hamid proposed a fluffy rationale based SDP model. The presentation of this rationale based forecast model has been checked by genuine programming projects information. They track down this model as the best method for acquiring prevailing arrangement of measurements. This thus make fluffy rationale based model more legitimate and good when contrasted with different models. Result showed that utilizing all product measurements gives the most minimal exactness and less fulfillment as contrasted and the other arrangement of measurements. The applicable arrangement of measurements gives better outcome that is measurements gotten after expulsion of excess measurements.

In [2], Koroglu et al. utilized seven old renditions of programming and their extra component to track down the deformities of current forms. They analyzed a few SDP process that is Naïve Bayes, choice tree, and irregular backwoods and observes the arbitrary woods has the most noteworthy prescient power when contrasted with different models. This multitude of models are contrasted and the AUC esteem that is region under bend. They observe that irregular timberland has the most elevated AUC esteem.

In [3], Sharmin proposed an original strategy of quality determination that is choice of trait with log sifting (SAL). They utilized the log sifting to preprocess the information. At long last, reaches the resolution that this strategy gives the more exactness of SDP when contrasted with different procedures. This technique is applied on a few broadly utilized openly accessible datasets

In [4], Sethi and Gagandeep track down that the fake brain organization (ANN) gives the better outcome when contrasted with fluffy based rationale model. ANN gives the more precise worth. It tends to be utilized in half breed way to deal with an enormous dataset. These model is investigated with the mean size of relative blunder (MMRE) and adjusted mean greatness of relative mistake (BMMRE).

In [5], Suffian involved the measurements to observe the exhibition of various models that is relapse model with different models. They observe that relapse investigation is generally exact when contrasted with different models. They utilized the p-worth of 0.05 as the edge for the choice of qualities of programming.

3. Methodology:

AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique which combines multiple "weak classifiers" into a single "strong classifier". It was formulated by Yoav Freund and Robert Schapire. They also won the 2003 Gödel Prize for their work.

Algorithm:

```
1. Initialise the dataset and assign equal weight to each of the data point.
2. Provide this as input to the model and identify the wrongly classified data points.
3. Increase the weight of the wrongly classified data points..
4. if (got required results)
5. Goto step 5
6. else Goto step 2
7. End
```

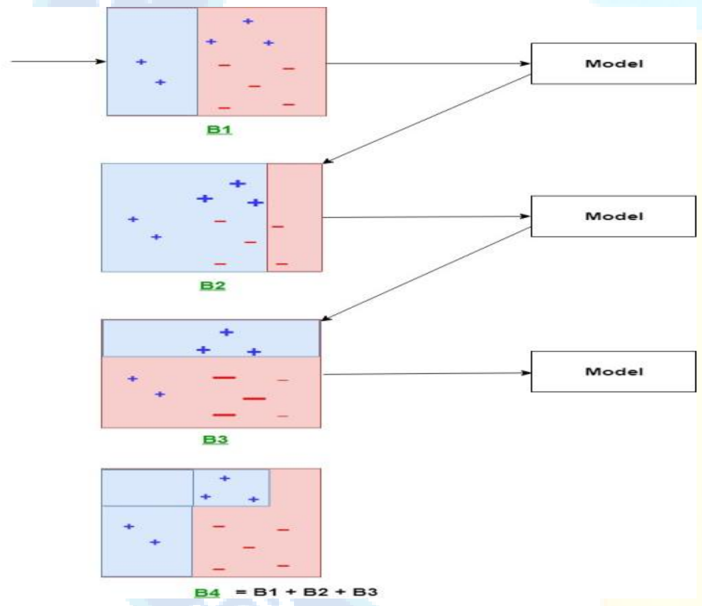

Fig. 1. Adaboost regression model.

4. Result and Discussion:

The proposed model is implemented by using the python language. Python is a dynamically typed oops based language. It is majorly used in the field of AI, deep learning, machine learning, data science and analytics, as well in field of web devolopment. The python provides various preprocessed api for the implemention of machine learning models. We have acquired 5 different open source dataset which we will use to devolop the software quality assurance model. The data set is orginally software failure data, which is made availabe by Musa (1979). The data set is of SPSS 14.0 availabe on http:// www.spss.com. The data set contains the cumulative failure time and inter failure time. the dataset is stored in comma seprated files which is read via the pandas (python data analysis) library. Then the data is visualized by using matplotlib library. The figure 2 shows the graph of the different data set used.

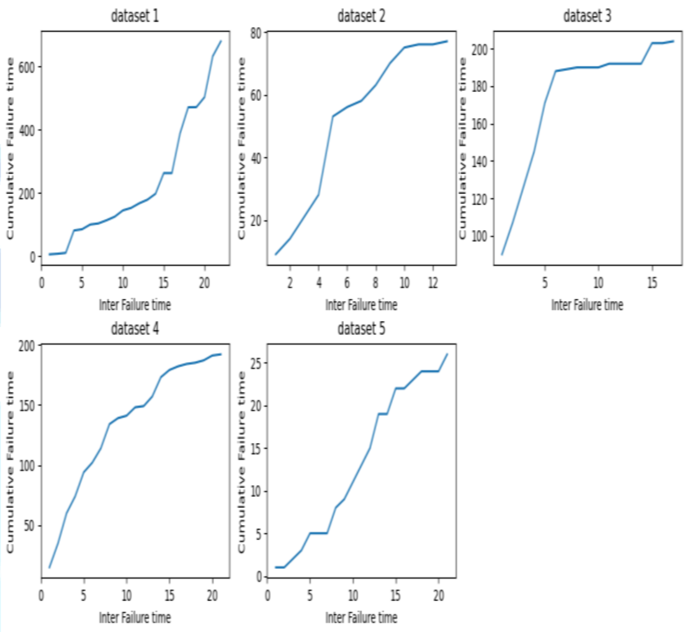

Fig. 2. Correlation of the Different Data Set

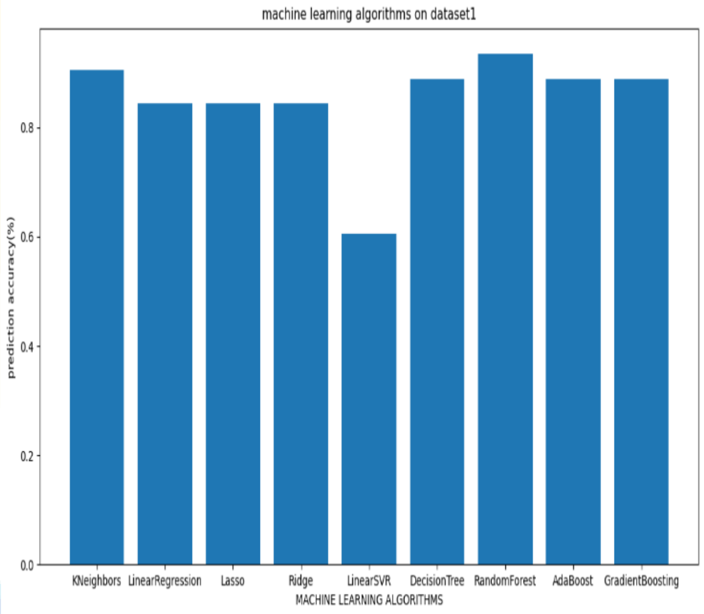

Fig. 3. Result of machine learning models on first data set

5. Conclusion:

Software reliability is the integral part of software quality. Software quality has been a major concern for the companies from past. The software reliability is unpredictable due to its probabilistic nature but with the help of machine learning models we have developed a prediction models for the prediction of software reliability. The ensemble models including both the bagging and boosting have outperformed all the other models. The ensemble learning models used were random forest, Adaboost, Gradientboost algorithms. The KNeighbors algorithm also have shown good results on the open source software reliability dataset. The SPSS 14.0 open source dataset was used to train the machine learning models. Software reliability changes frequently from software to software very quickly hence it is very difficult to predict the software reliability. We can improve the software reliability prediction by using the proposed methodology and a proper pre-processing of the dataset.

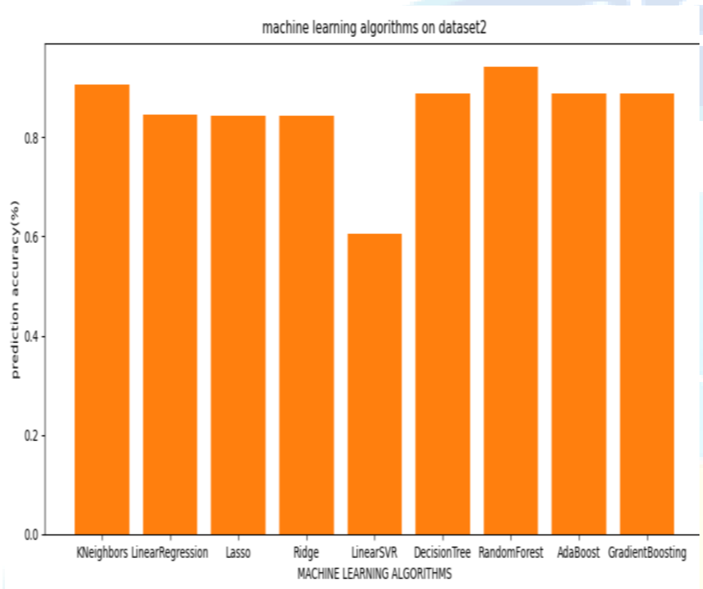

Fig. 4: Result of machine learning models on second data set

References:

[1]. Ai-jamimi, H. A. (2016). Toward comprehensible software defect prediction models usingfuzzy logic (pp. 127–130).
[2]. Koroglu, Y., Sen, A., Kutluay, D., Bayraktar, A., Tosun, Y., Cinar, M., & et al. (2016). Defectprediction on a legacy industrial software : A case study on software with few defects. In 2016IEEE/ACM 4th International Workshop on Conducting Empirical Studies in Industry (CESI)(pp. 14–20).
[3]. Sharmin, S. (2015). SAL: An effective method for software defect prediction (pp. 184–189).
[4]. Sethi, T., &Gagandeep. (2016). Improved approach for software defect prediction usingartificial neural networks. In 2016 5th International Conference on Reliability, InfocomTechnologies and Optimization (Trends and Future Directions) (pp. 480–485).
[5]. Suffian, M. D. M., Ibrahim, S., Dhiauddin, M., Suffian, M. D. M., & Ibrahim, S. (2012).A prediction model for system testing defects using regression analysis. International Journalof Soft Computing and Software Engineering, 2(7), 69–78.
[6]. Mandal, P.,&Ami, A. S. (2015). Selecting best attributes for software defect prediction. In 2015IEEE International WIE Conference on Electrical and Computer Engineering (pp. 110–113).
[7]. Can, H., Jianchun, X., Ruide, Z., Juelong, L., Qiliang, Y., &Liqiang, X. (2013). A new modelfor software defect prediction using Particle Swarm Optimization and support vector machine.In 2013 25th Chinese Control and Decision Conference (pp. 4106–4110).
[8]. Jiarpakdee, J., Tantithamthavorn, C., Ihara, A., & Matsumoto, K. (2011). A study ofredundant metrics in defect prediction datasets (pp. 37–38).
[9]. Wang, T.,&Li,W. (2010).NaïveBayes software defect predictionmodel. IEEE, no. 2006 (pp. 0–3).
[10]. Liu, J., Xu, Z., Qiao, J., & Lin, S. (2009). A defect prediction model for software based onservice oriented architecture using EXPERT COCOMO. In 2009 Chinese Control andDecision Conference (pp. 2591–2594).
[11]. Kakkar, M., & Jain, S. (2016, January). Feature selection in software defect prediction: Acomparative study. In 2016 6th International Conference on Cloud System and Big DataEngineering (Confluence), (pp. 658–663).
[12]. Verma, D. K., & Kumar, S. (2015). Emperical study of defects dependency on softwaremetrics using clustering approach (pp. 0–4).
[13]. Yang, X., Tang, K., & Yao, X. (2015). A learning-to-rank approach to software defectprediction. IEEE Transactions on Reliability, 64(1), 234–246.
[14]. Sawadpong, P., & Allen, E. B. (2016). Software defect prediction using exception handlingcall graphs : A case study.
[15]. Shuai, B., Li, H., Li, M., Zhang, Q., & Tang, C. (2013). Software defect prediction usingdynamic support vector machine.In 2013 9th International Conference on ComputationalIntelligence and Security (CIS) (pp. 260–263).
[16]. Armah, G. K., Luo, G., & Qin, K. (2013). Multi_level data pre_processing for software defectprediction.In 2013 6th International Conference on Information Management, InnovationManagement and Industrial Engineering (ICIII) (pp. 170–174).