International Journal of Art & Humanity Science (IJAHS)
www.ijahs.com Volume 3 Issue 5, (Sept-Oct 2016), PP.30-40

Software Defect Prediction Using Data Mining Classification Approach

Vinay kumar Dwivedi vinaydwived@gmail.com

Mahesh Kumar Singh Assistant Professor Bansal Institute of Engineering & Technology,

Lucknow mks.cse07@gmail.com

Abstract

Defect in software systems continue to be a major problem. High quality of software is ensured by Software reliability and Software quality assurance. A software defect causes software failure in an executable product. A variety of software fault predictions techniques have been proposed, but none has proven to be consistently accurate. The objective in the construction of models of software error prediction is to use measures that may be obtained relatively early in the software development life cycle to provide reasonable initial estimates of quality of an evolving software system. Here various data mining classification and prediction techniques viz. Neural Network (NN), Naïve Bayes, k-Nearest Neighbour (kNN) have been analysed and compared for software defect prediction model development. For this DATATRIEVETM project carried out at Digital Engineering, Italy has been used to validate the algorithm. The results showed that model using NN classification technique was a better prediction model.

Keywords: Software Defect, NN, kNN, Naive Bayes, Classification techniques, Data Mining

Introduction

Faults in software systems continue to be a major problem [1]. They are present in a computer program as errors, flaws, defects, failures, or faults. This hinders the software from working in the desired manner. (e.g., a faulty result being produced) [2]. A software fault is a defect that causes software failure in an executable product. Number of defects in a module of a software can be effectively identified using Software metrics-based quality prediction models as a tool. The use of such models before every planned release of the product, or deployment of that may considerably improve system quality [3]. A defective prediction model is identified using metrics from an earlier deployment or identical projects, and is then implementd in modules presently under development. Afterwards, a timely prediction of that modules needs lots of effort to get rid of the defects and then it can be secured. Over the past decades years, several empirical studies have been carried out to predict the fault proneness models. Software fault prediction study can be grouped as statistical and machine learning (ML) technique, of which the machine learning technique is the most popular [4]. Unluckily, the seriousness of software fault prediction have not resolved methodically. And none of the techniques have achieved widespread applicability in the software industry due to several reasons, including the limitation of testing resource, absence of software tools to mechanize this software fault prediction, the unwillingness to collect the software defect data, lots of method based on the private software data, and the other practical problems.

Machine learning classification algorithm is an accepted technique for software fault prediction [6]. Classification forecasting has two levels: classifier construction and the usage of the classifier constructed. The former is concerned with the building of a classification model. Here it deals with a set of preset classes using training dataset. Here in the training data, all the samples are thought of as belonging to a preset class. This is determined by the class attribute label. The model so developed is designated as a classification rules, decision tree or mathematical formula.

In the current work, a relative analysis of a variety of classification techniques has been proposed for getting better performance of software defect prediction. It is seen that particle swarm optimization is useful for feature selection, and bagging one for class imbalance problem. Bagging technique is useful in managing class imbalance. The current work is carried out using public datasets from DATATRIEVETM project carried out at Digital Engineering, Italy.

Literature Review

Various techniques, such as linear regression, discriminate analysis, decision trees, neural networks etc. have been developed and applied to predict defects in software. Ahmet Okutan, et.al.(2012), proposed a novel method using Bayesian networks to explore the relationships among software metrics and defect proneness. Mrinal Singh Rawat et. al.(2012), identified causative factors which in turn suggest the remedies to improve software quality and productivity. Yajnaseni Dash, Sanjay Kumar Dubey, (2012) aimed to survey various research methodologies proposed to predict quality of OO metrics by using neural network approach. Ms. Puneet Jai Kaur, Ms. Pallavi, (2013) discussed data mining techniques that are association mining, classification and clustering for software defect prediction. Sonali Agarwal and Divya Tomar, (2014), proposed a feature selection based Linear Twin Support Vector Machine (LSTSVM) model to predict defect prone software modules. Mrs.Agasta Adline, Ramachandran. M(2014) Predicting the fault-proneness of program modules when the fault labels for modules are unavailable is a challenging task frequently raised in the software industry. Pooja Paramshetti , D. A. Phalk, (2015), applied association rule discovery for detecting software entities that are likely to be defective in software systems. H. S.

Shukla, Deepak Kumar Verma (2015), analysed various literatures on defect prediction and drew various conclusions.

Data Used

In the present paper various data mining classification algorithms will be applied for the development of an efficient predictive model. For this DATATRIEVETM project carried out at Digital Engineering, Italy has been used to validate the algorithm [8]. It has 130 records, which includes total nine attributes, of which eight are condition attributes and one is decision attribute. They are taken from open source PROMISE Software Engineering Repository data set with the intention of making available these datasets for the advancement of research in the field of software engineering using different model development techniques. This one includes a linguistic attribute (DEFECTS) to indicate defectiveness. It is given in 0/1, which indicates no faults / faults found. Hence for the development of the classification models these values has been converted into NO/YES as label attributes. The descriptions of the features are taken from http://promise.site.uottawa.ca/SERepository/datasets/datatrieve.arff [5].

Table 1 Input and Output model parameters used for model development

Input Variable	@attribute LOC6_0 numeric @attribute LOC6_1 numeric @attribute Added_LoC numeric @attribute Del_LoC numeric @attribute Diff_Block numeric @attribute Mod_Rate numeric @attribute Mod_Know numeric @attribute ReusedLoC numeric
Output Variable	@attribute Faulty6_1 {0, 1}

Problem Statement and Proposed Technique

This section presents the proposed technique to analyze software defect data. The proposed approach uses permutation combination of various classification techniques, viz. Neural

Network (NN), Decision tree (DT), Support Vector Machine (SVM), K-Nearest Neighbour (k-NN)**.** Further stacking, a meta modeling techniques in order to enhance the accuracy of the classification techniques have also been used in the model. In the first stage a pre-processing model is proposed to optimize the dataset. In the second stage experiments are performed using the machine learning classification methods to obtain the performance vector for various software fault prediction models. The proposed framework is depicted in figure 1.

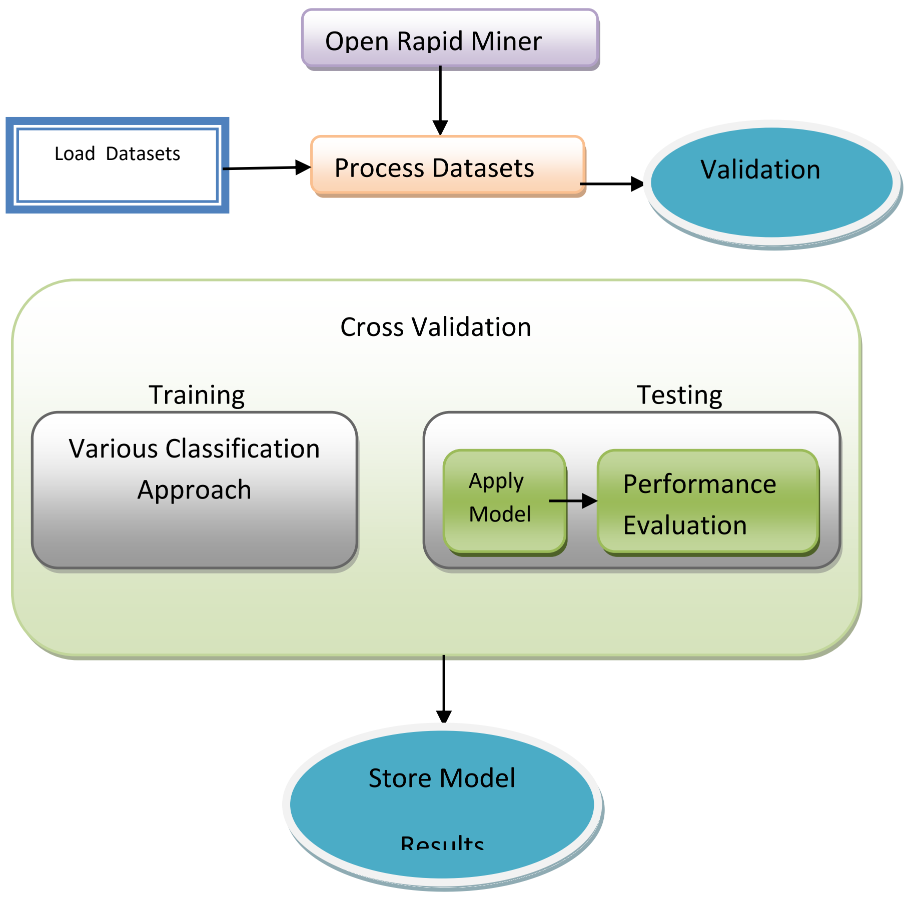


Figure 1: Experimental Framework for Data Analysis

Model Development Stages

In the present work two software error prediction models have been developed using Rapid Miner tool. The details of the two models using different algorithms are tabulated below.

Table 2: Algorithms used for MODEL-I and MODEL-II

Mode l	Main Process	Basic Learner	Stacking Model Learner	Training Algorithm	Learning Process
I	X-Validation	***	***	NN	***

II	X-Validation	k-NN, NN	Naive Bayes	Stacking	***

k-NN= k Nearest Neighbourhood, NN=Neural Network,

The flowchart used in Rapid Miner for carrying out software error prediction model development are given in Figures 2 and 3 below,

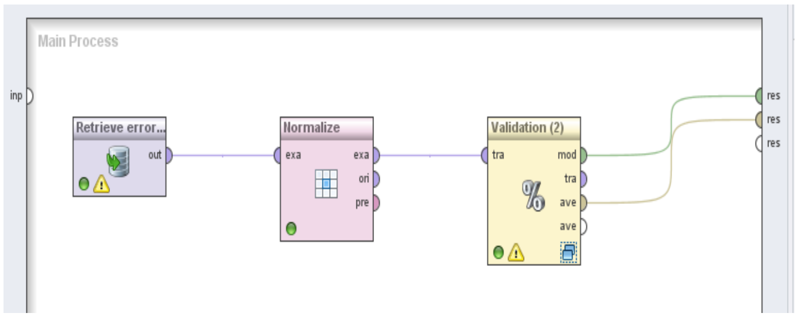


Fig 2 : Screen Shot of Main Process of MODEL-I

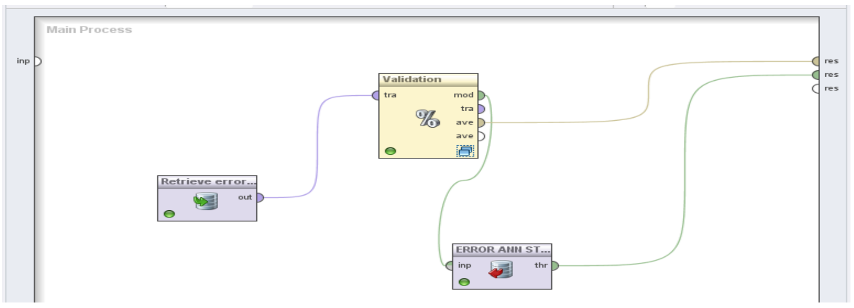


Fig 3 : Screen Shot of Main Process of MODEL-II

The various stages include data retrieve, normalization of data, data validation, which includes training and testing sub-process. In Model-I training is done using NN classifier, where as in Model-II training includes stacking meta-learner. The Stacking operator is a nested operator, having two sub-processes, the base learner and the Stacking model learner. The base learner uses kNN and NN whereas stacking model learner uses Naive Bayes. Next, testing incorporates Apply model and Performance Evaluation. The **Apply Model** operator applies the already learnt (trained) model on an ExampleSet. The **Performance** operator is used for performance evaluation, and delivers a list of performance criteria values. These performance criteria are automatically determined in order to fit the learning task type.

Table 3 : Parameter Used for Model Development

Sl. No.	Operator		Parameter Used	Type
	MODEL - I			
1	Normalize	**a**	Min. val.	0

	Using range transformation	b	Max. Val.	**1**
2	Validation	a	no. of validations	10
		b	sampling type	Automatic
3	Neural Net	a	criterion	Training cycle – 500
				Learning rate – 0.5
				Momentum – 0.9
4	Performance	a	Accuracy	
		b	Precision	
		c	Recall	
		d	RMSE	
		e	Absolute error	
	MODEL - II			
1	Validation	a	no. of validations	10
		b	sampling type	Automatic
2	k-NN	a	k	1
		b	Measure Type	Mixed Measure using Euclidean distance
3	Neural Net	a	criterion	Training cycle – 500
				Learning rate – 0.3
				Momentum – 0.2
4	Naïve Bayes	a	Laplace Correction	

Results and Discussions:

The dataset consists of one includes a linguistic attribute (DEFECTS) to indicate defectiveness. It is given in 0/1, which indicates no faults / faults found. Hence for the development of the classification models these values has been converted into NO/YES as label attributes.

If an attribute is labeled as yes and is classified as yes it is counted as true positive else if it is classified as no it is counted false negative. Similarly, if a label is labeled no and is classified as no it is counted as true no else if it is classified as yes it is counted as false no. Based on these outcomes a two by two confusion matrix can be drawn for a given test set. This is shown in Figure 4.8 and 4.9 below for both the models I and II.

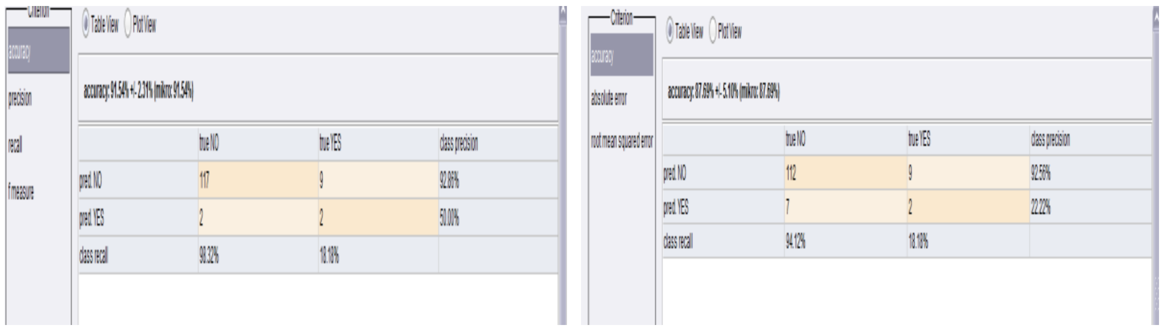
The confusion matrix in figure 4 below forms the basis for the calculation of the following metrics.

i. *Accuracy* = (tp+tn)/ (P+N)

ii. *Precision* = tp/ (tp+fp)

iii. *Recall/ true positive rate* = tp/P

iv. *F-measure* =2/ ((1/precision)+(1/recall))



accuracy: 91.54% +/- 2.31% (mikro: 91.54%)

	true NO	true YES	class precision
pred. NO	117	9	92.86%
pred. YES	2	2	50.00%
class recall	98.32%	18.18%	

accuracy: 87.69% +/- 5.10% (mikro: 87.69%)

	true NO	true YES	class precision
pred. NO	112	9	92.56%
pred. YES	7	2	22.22%
class recall	94.12%	18.18%	

Fig. 4 : Confusion Matrix for MODEL- I &II

The accuracy of the models I and II are 91.54 and 87.69 and their graphical representation are shown in Fig. 5 below.

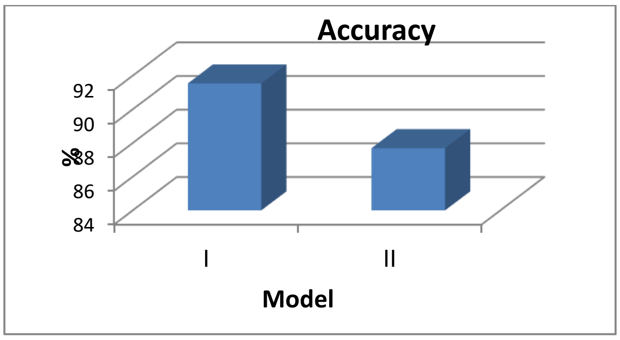


Fig. 5: Graphical Representation of Model Accuracy

Further below is given the tabulation of Absolute error, RMSE, Recall, Precision and f-measure of both the ModelsI and II.

Table 4: Error Values of Model – I & II

Model No.	Abs. Error	RMSE
I	0.124	0.276
II	0.126	0.338

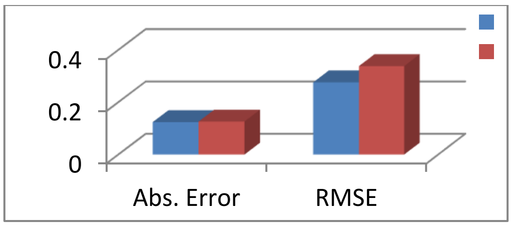


Fig. 6 : Comparative Plot of AE and RMSE OF MODEL-I and MODEL-II

Table – 5

Model No.	Precision	Recall	f-Measure
I	50	15	26.67
II	22.22	15	20

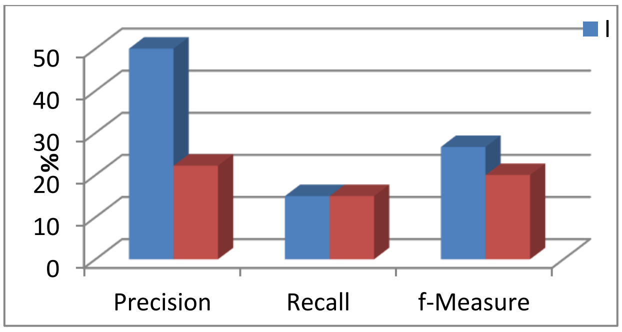


Fig. 7 : Comparative Plot of MODEL-I and MODEL-II

From the above analytical study of Table 4 and 5 and their respective graphs in Fig. 5, 6 and 7 as regards the performance analysis of both the MODEL –I and MODEL-II, it can be seen that MODEL-I has a prediction accuracy of 91.54% a compared to that of MODEL-II with 87.69% accuracy. The RMSE and AE values are also better for MODEL-I as compared to MODEL-II. Although Recall for both the models are almost same, but Precision and f-measure for MODEL-I is better than MODEL-II. Thus MODEL-I using Neural Network training algorithm has been able to develop a better model as compared to MODEL-II using Stacking as training algorithm .

Conclusion

Software developers and quality control managers must come out with a variety of combinations like persons, tools, development techniques, etc. so as to be able to develop quality products and be able to deliver it on time, that too within budgetary cost. Thus in order to tackle the above entioned issues. An attempt has been made in the present work for software error prediction. Here various data mining classification and prediction techniques viz. Neural Network (NN), Naïve Bayes, k-Nearest Neighbour (kNN) have been analysed and compared for software defect prediction model development. For this DATATRIEVETM project carried out at Digital Engineering, Italy has been used to validate the algorithm. It has 130 records, which includes total nine attributes, of which eight are condition attributes and one is decision attribute.

Two model MODEL-I and MODEL-II were developed and compared. MODEL-I was developed using NN training algorithm and was found to be a better prediction model as compared to MODEL-II which used stacking as training algorithm. MODEL-I had an accuracy of 91.54% as compared to MODEL-II with 87.64% accuracy. Thus it can be concluded that Neural Network training algorithm is a better classification tool for the development of software prediction model than as compared to stacking model, using Naïve Bayes as Stacking model learner and k-NN and Neural Net as Base Learner.

References:

1. Parvinder S. Sandhu, Sunil Khullar, Satpreet Singh, Simranjit K. Bains, Manpreet Kaur, Gurvinder Singh, "A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm", World Academy of Science, Engineering and Technology, 2010, pp. 648-653.
2. http://puretest.blogspot.com/2009/11/1.html
3. Ahmet Okutan, et. al., (2012), "Software defect prediction using Bayesian networks", Empir Software Eng (2014) 19:154–181
4. Mrinal Singh Rawat, et. al.,(2012), "Software Defect Prediction Models for Quality Improvement: A Literature Study", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2,
5. DATATRIEVETM project DATABAASE, http://promise.site.uottawa.ca/SERepository/datasets/datatrieve.arff .
6. Jang, J-S. R., (1993), *"ANFIS-Adaptive-Network Based Fuzzy Inference System"*, IEEE Transactions on Systems, Man and Cybernatics, 23(3), pp 665-685.
7. Sonali Agarwal and Divya Tomar, (2014), " A Feature Selection Based Model for Software Defect Prediction", International Journal of Advanced Science and Technology Vol.65 (2014), pp.39-58.
8. Romi Satria Wahono and Nanna Suryana (2013), "Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction", International Journal of Software Engineering and Its Applications Vol.7, No.5 (2013), pp.153-166.
9. Ahmet Okutan, Olcay Taner Yıldız,(2012) "Software defect prediction using Bayesian networks", Empir Software Eng (2014) 19:154–181 © Springer Science+Business Media, LLC.

10. Mrinal Singh Rawat, Sanjay Kumar Dubey,(2012) "Software Defect Prediction Models for Quality Improvement: A Literature Study", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, pp 288-296.
11. Yajnaseni Dash, Sanjay Kumar Dubey, (2012), " Quality Prediction in Object Oriented System by Using ANN: A Brief Survey", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 2,, pp.1-6.
12. Ms. Puneet Jai Kaur, Ms. Pallavi, (2013), " Data Mining Techniques for Software Defect Prediction", International Journal of Software and Web Sciences (IJSWS), International Journal of Software and Web Sciences 3(1), pp. 54-57.
13. Sonali Agarwal and Divya Tomar, (2014), " A Feature Selection Based Model for Software Defect Prediction", International Journal of Advanced Science and Technology Vol.65 (2014), pp.39-58.
14. Mrs.Agasta Adline, Ramachandran. M(2014), "Predicting the Software Fault Using the Method of Genetic Algorithm", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 2,, pp 390-398.
15. Pooja Paramshetti, D. A .Phalk, (2015), "Software Defect Prediction for Quality Improvement Using Hybrid Approach", *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, Volume 4, Issue 6, June 2015, pp.99-104.
16. H. S. Shukla, Deepak Kumar Verma (2015), "A Review on Software Defect Prediction", *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 12, pp. 4387-4394.*