International Journal of Computer Science and Software Engineering
Volume 1, Number 1 (2015), pp. 39-48

http://www.irphouse.com

Software Defects Prediction Using Support Vector Machine

Rishu Gupta

Student, Department of CSE, Lovely Professional University
Phagwara, Punjab, (India) - 144411
rshgrg0@gmail.com

Abstract

In the field of software engineering, there are a number of prediction methods that are fault estimation, security estimation, effort estimation, correlation cost estimation, re-usability estimation, test effort estimation and quality estimation. These methods are helps to reduce the cost of testing which ultimately reduce the cost of the project. One of the most important stage is testing of software which reduces the defects of the software. Software defects are predicted by a software developers or testers at an early stage of software development life cycle and it reduces the overall time, cost and effort of the software development team. Nowadays, defect prediction methods are based on an adequate quantity of historic project data. Many researchers are using NASA's PROMISE dataset to develop prediction techniques for various phases of software development. We use the dataset available for defect prediction. This work aims to achieve high prediction accuracy by applying Support Vector Machine based technique. In this research work, our primary target will be to focus on making a MATLAB based interface to predict software defects. This interface will implement SVM as the underlying algorithm and will be trained and tested using ANT-1.7 dataset.

1. Introduction

Software metrics are utilized to discover the Software faults in the Software improvement life cycle before the testing procedure. Routines utilized are insights, machine learning, machine adapting alongside measurable techniques and factual models versus expert estimation [1]. Defects in programming frameworks prompts significant issues in software. A large portion of programming frameworks are conveyed to clients with intemperate issues. To discover the defect-prone parts of the software and to focus on those parts for expanded quality control and testing is a

compelling way to deal with decrease of faults. There are different characteristics like deformity thickness, viability, issue inclination, standardized revamp and re-convenience which focus the nature of the product.

Support Vector Machine (SVM) is introduced in COLT-92 by Boser, Guyon&Vapnik. It is theoretically a very well motivated algorithm. Vapnik&Chervonenkis (1960s) developed SVM from Statistical Learning Theory. In SVM, information is being differentiated into two sets; training set and testing set. Every record in the training set contains one target value or class name and contains a few properties known as watched variables. SVM discovers a direct dividing hyper-plane. The equation for partition is ax+by=c. SVM is utilized as a part of numerous fields. SVM is utilized as a part of twofold arrangement errands. SVMs are another promising non-direct, non-parametric order method. SVM is utilized as a part of the restorative diagnostics, optical character recognition, electric load anticipating and other numerous fields.

In this paper, we propose a multi-class SVM based technique to predict software defects. The proposed technique has been implemented into a simple yet effective graphical user interface to let the user easily input the required parameters and predict the number of software defects that may arise in the future. This detection leads to a very convenient phase in the software development life cycle, where the user can make necessary changes in the software and/ or the methodology being applied so that the faults can be avoided at later stages. Hence, it can save a huge sum in the overall development cost of the software. For training and testing purposes, dataset from NASA namely ANT 1.7 has been used. The detail of the dataset has been given in the forthcoming sections of this paper. Experiments have shown a very encouraging 89.0909% of accuracy in the prediction results.

The rest of the paper has been divided into 5 sections. Section 2 presents review of literature. Section 3 explains proposed methodology including the insight to dataset. Section 4 deals with implementation. Section 5 gives experimental results and section 6 concludes the findings.

2. Literature Review

To predict the Software defects, researchers can also apply a machine learning approach on real-time Software systems. Examples include telecontrol/ telepresence, robotics and mission planning systems. There are number of prediction techniques that are used like (Statistical models such as Stepwise Multi-linear Regression models and multivariate models, and machine learning models, such as Artificial Neural Networks, Instance-based Reasoning, Bayesian-Belief Networks, Decision Trees and Rule Induction) but for all thedata sets there is no such a technique that gives accurate result.

Fenton et al. have presented a dataset for only 31 software development projects [2]. This datasetincorporates the set of quantitative and qualitative factors that were previously built into a causal model of the software process. The factors include values for code size, effort and defects, together with qualitative data values judged by project managers using a questionnaire. They have used these data to evaluate the

causal model. They claim that good predictions of the defects can be achieved byentering relatively few of the project factors.

Design [3] and code metrics [4] are used to compare the accuracy of fault prediction models that are available before and after the system is implemented. Code metrics and design metrics are available only after the system is implemented and before the coding has started [5]. Models are based on the data from one release of a large telecommunication system developed by Ericsson using linear regression. In their study, prediction made after the system is 34% more accurate than before the system. The variability of metrics available before the implementation is 43% and after the implementation is 58% [5] but the performance of the system is same when metrics are not used.

Specialists use Statistical strategies and Machine learning methods to anticipate the shortcoming inclination of the code in their software. In their study, execution of Lines of code (LOC) metric is well and accuracy of Lack of Cohesion on Methods (LCOM) metric is great however its culmination quality is low. For fine grained examination multivariate models perform better [6].

Song et al. propose and evaluate a general framework for software defect prediction that supportsunbiased andcomprehensive comparison between competing prediction systems [7].

Koru and Liu demonstrate that static measures and defect data collected at class level can be used to build machine-learning models that predict top defect classes in practice [8].

Okutan and Yıldız use Bayesian networks to determine the probabilistic influential relationships among software metrics and defect proneness. In addition to the metrics used in Promise data repository, they define two more metrics, i.e. NOD for the number of developers and LOCQ for the source code quality. They extract these metrics by inspecting the source code repositories of the selected Promise data repository data sets [9].

Hierarchical, k-means clustering and neural network was used to find groups of similar projects [10]. The obtained clusters were investigated with the discriminant analysis. For each of the identified group a statistical analysis has been conducted in order to distinguish whether thisgroup really exists. Two defect prediction models were created for each of the identified groups. The first one was based on the projects that belong to a given group, and the second one - on all the projects.

Li et al. describe three methods for selecting a sample: random sampling with conventional machine learners, random sampling with a semi-supervised learner and active sampling with active semi-supervised learner [11]. To facilitate the active sampling, they propose a novel active semi-supervisedlearning method ACoForest which is able to sample the modules that are most helpful for learning a good prediction model.

Bibi et al. apply a machine learning approach to the problem of estimating the number of defects calledRegression via Classification (RvC) [12]. RvC initially automatically discretizes the number of defects into a number of fault classes, then learns a model that predicts the fault class of a software system. Finally, RvC transforms the class output of the model back into a numeric prediction.

Fenton et al. use graphical probability models, known as Bayesian Belief Networks as the appropriate formalism for representing this evidence [13]. They use the subjective judgements of experienced project managers to build the probability model and use this model to produce forecasts about the softwarequality throughout the development life cycle. The causal or influence structure of the modelmore naturally mirrors the real world sequence of events and relations than can be achieved with otherformalisms. Their work focuses on the particular model that has been developed for Philips Software Centre (PSC), using expert knowledge from Philips Research Labs.

Challagulla et al. evaluate different predictor models on four different real-time software defect datasets [14]. Their results show that a combination of 1R and Instance-based Learning along with the Consistency based Subset Evaluation technique provides a relatively better consistency in accuracy prediction compared to other models. They also claim that size and complexity metrics are not sufficient for accurately predicting real-time software defects.

Ratzinger et al. analyze the influence of evolution activities such as refactoring on software defects [15]. In a case study of five open source projects they used attributes of software evolution to predict defects in time periods of six months. They use versioning and issue tracking systems to extract 110 data mining features, which are separated into refactoring and non-refactoring related features. These features are used as input into classification algorithms that create prediction models for software defects.

3. Proposed Methodology

We propose the use of Multi-class Support Vector Machine (McSVM) to classify the input set of parameters into one of the several classes to predict the number of faults that may arise in software that is under construction. Figure 1 shows classification in case of multi class SVM.

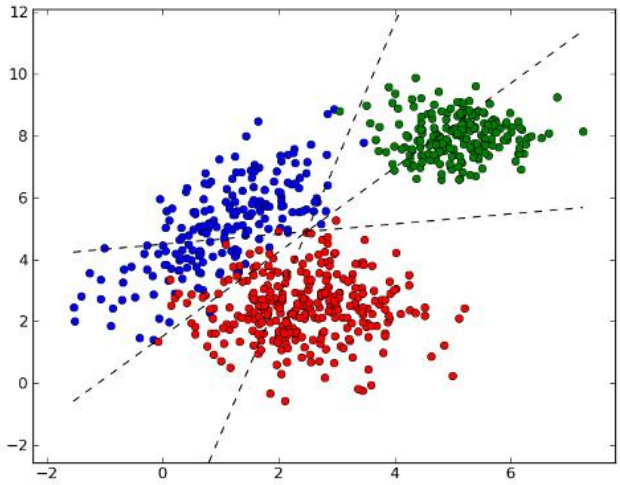

Fig. 1: Classification in Multi Class SVM.

The McSVM implementation uses an "all-together" multi-classification approach, which is computationally more expensive yet usually more accurate than "one-against-all" or "one-against-one" multi-classification methods. Comparisons of these methods using large-scale problems have not been seriously conducted. Especially for methods solving multi-class SVM in one step, a much largeroptimization problem is required so up to now experiments are limited to small data sets.

In the present research work, we use ANT 1.7 dataset given by NASA's Promise dataset repository. The dataset contains 745 records. We divided the dataset into two subsets; training set containing 635 records (approx. 85% of total) and testing dataset containing 110 records (approx. 15% of total). These two datasets are used to train and test the tool, respectively. Each record of the dataset provides us with 18 input metrics and 1 target or output parameter. These metrics are explained in detail by Chidamber and Kemerer [16]. Table 1 lists all the metrics used.

Table 1: Metric names and abbreviations used in ANT 1.7 dataset.

Sr. No.	Metric Name	Abbreviation used
1	Weighted methods per class	WMC
2	Depth of Inheritance Tree	DIT
3	Number of Children	NOC
4	Coupling between object classes	CBO
5	Response for a class	RFC
6	Lack of cohesion in methods	LCOM
7	Afferent Couplings	Ca
8	Efferent Coupling	Ce
9	Number of public methods	NPM
10	Lack of cohesion in methods	LCOM3
11	Lines of Code	LOC
12	Data access metric	DAM
13	Measure of Aggregation	MOA
14	Measure of Functional Abstraction	MFA
15	Cohesion Among Methods of Class	CAM
16	Inheritance Coupling	IC
17	Coupling Between Methods	CBM
18	Average Method Complexity	AMC

4. Implementation

4.1 User Friendly Tool

A user friendly tool using MATLAB R2015a has been developed. This tool allows the user to train the McSVM. After the training is over, it lets the user input various parameter values to predict the number of faults that may arise in the software being developed. To test the accuracy of the proposed system, an option has been given

which reads the test dataset and compares the predicted and actual outputs. Figure 2 shows the GUI with various input boxes and buttons to train, predict and test.

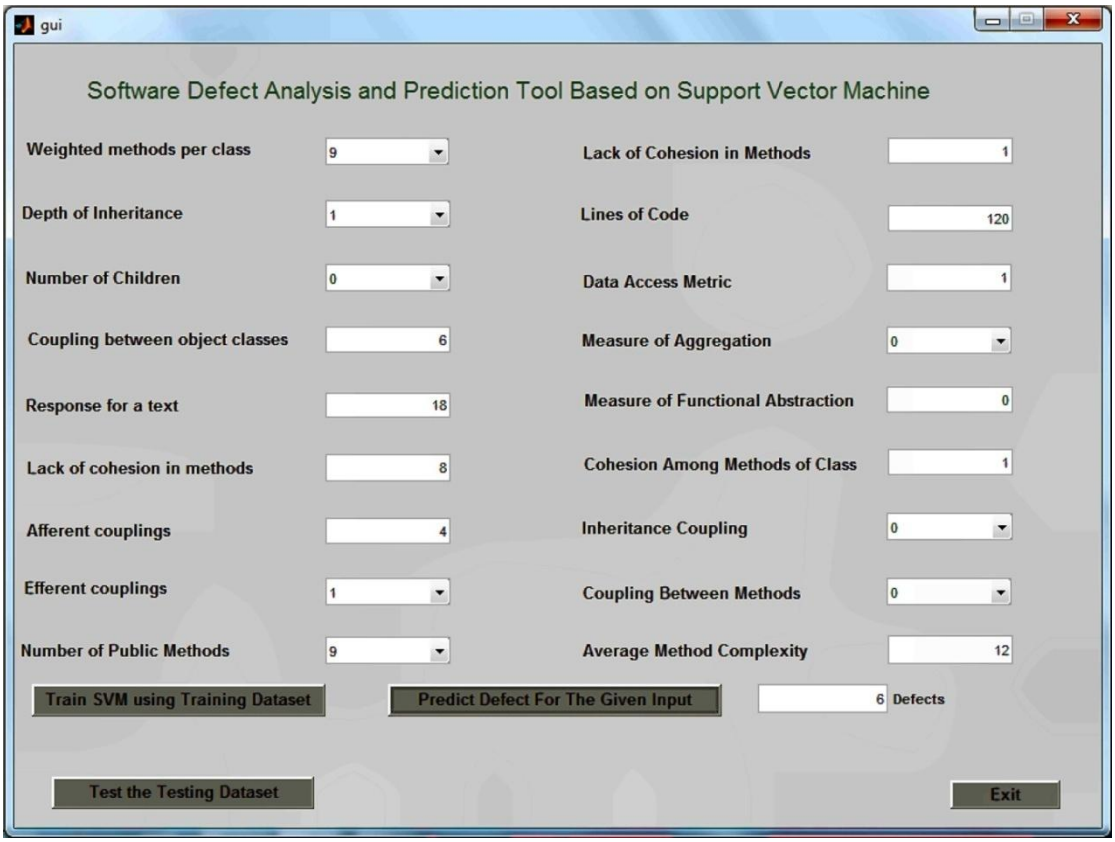

Fig. 2: User Friendly GUI.

4.2 Preparing the data

Data required to train and test the system has been taken from the PROMISE repository. We chose ANT-1.7 dataset. This dataset contains a total of 745 records, out of which 635 (nearly 85%) were used to train the McSVM and rest (110, nearly 15%) were used for testing the system. The selection of record for training and testing was done on a pure random basis.

4.3 Train and then predict the Multiclass SVM

It is necessary to train the McSVM prior to making any use of it. Thus, an option is provided on the panel to train McSVM. There is a file namely, traindata.txt available for the tool to read and get trained. It reads all the 635 training records one by one and accordingly gets trained. Once the McSVM training is finished, the tool is ready for prediction or testing. It is wise to test the tool before deploying it for prediction. Thus, we use the option to test the testing dataset. Again, there is a file namely, testdata.txt available. The tool reads all the records one by one, applies the McSVM classification process and gets the results. These results are then compared with the actual outputs

given in each record of the test dataset. Finally, when the authenticity of the tool has been established, it can be put to actual use of prediction. The user may enter the values using textboxes and drop down boxes and can get the predicted result on the panel itself.

5. Experimental Results

Experiments have shown that the prediction accuracy of the proposed technique is quite encouraging. We are able to achieve an accuracy of 89.0909%. As compared to the proposed technique, the accuracy of one of the previously available work is given in table 2. The highest accuracy that it achieved is 71.9%, which is far below the accuracy achieved by the proposed technique.

Table 2: Accuracy of technique given in [17].

Technique used	Dataset	Accuracy percentage
Genetic Algorithm with SVM	jE1	71.9%
	jE2	63.7%
	jE1&jE2 (inter-release)	58.0%

Table 3 shows the results of testing the test dataset. The last two columns of the table show the predicted number of defects and actual number of defects, respectively.

Table 3: Comparison of Predicted and Actual Defects in Proposed Technique.

Sr. No.	Defects		Sr. No.	Defects	
	Predicted	Actual		Predicted	Actual
1	0	0	56	0	0
2	0	0	57	0	0
3	0	0	**58**	**4**	**3**
4	0	0	59	0	0
5	0	0	60	0	0
6	0	0	61	0	0
7	**1**	**0**	62	0	0
8	0	0	63	0	0
9	0	0	**64**	**2**	**0**
10	0	0	65	0	0
11	0	0	66	0	0
12	0	0	67	0	0
13	0	0	68	5	5
14	0	0	69	0	0
15	**1**	**0**	70	0	0

16	0	0	71	0	0
17	0	0	72	0	0
18	0	0	73	0	0
19	1	1	74	0	0
20	0	0	75	1	1
21	0	0	76	0	0
22	0	0	77	4	4
23	0	0	78	0	0
24	0	0	**79**	**7**	**5**
25	0	0	80	0	0
26	0	0	81	0	0
27	0	0	82	0	0
28	0	0	83	2	2
29	0	0	84	0	0
30	0	0	85	0	0
31	1	1	86	0	0
32	0	0	87	0	0
33	0	0	88	0	0
34	3	3	89	1	1
35	0	0	90	0	0
36	0	0	91	0	0
37	1	1	92	0	0
38	0	0	93	1	1
39	0	0	94	1	1
40	0	0	**95**	**1**	**0**
41	0	0	96	0	0
42	0	0	97	2	2
43	0	0	98	0	0
44	1	1	99	0	0
45	0	0	100	0	0
46	0	0	101	3	3
47	0	0	102	0	0
48	1	1	103	0	0
49	**3**	**4**	104	0	0
50	0	0	105	0	0
51	0	0	**106**	**1**	**0**
52	0	0	107	0	0
53	0	0	**108**	**3**	**4**
54	0	0	109	2	2
55	1	1	**110**	**3**	**2**

It is observed that in most of the cases, the technique has been accurately able to predict the number of defects. In some of the cases, the results are not as expected.

The differences in predicted and actual number of defects are shown in bold. There are a total of 12 differences. Thus, we calculate the error and accuracy percentage as follows:

$$Percentage\ Error = \frac{|Actual\ Outputs - Predicted\ Outputs|}{Actual\ Outputs} X100$$

$$Percentage\ Accuracy = 100 - Percentage\ Error$$

Using the above two formulae, we calculate the percentage error as 10.9091% and percentage accuracy as 89.0909%.

6. Conclusion

There can be a number of defects in software. To improve software quality, it is fundamental to predictthe software defects at an early stage of software development. The main objective of this paper is to study and understand the concept of software defects prediction using the multiclass support vector machine. In this study, we have used software metrics. This paper shows that the accuracy of the proposed system is 89.0909%, which is quite encouraging. In future, one can use other training algorithms clubbed with the proposed technique to further increase the accuracy level for predicting the software defects.

References

[1] Catal, C., &Diri, B. (2009). A systematic review of software fault prediction studies. Expert Systems with Applications, 36(4), 7346-7354.

[2] Fenton, N., Neil, M., Marsh, W., Hearty, P., Radlinski, L., & Krause, P. (2007, May). Project data incorporating qualitative factors for improved software defect prediction. InProceedings of the Third International Workshop on Predictor Models in Software Engineering(p. 2). IEEE Computer Society.

[3] El Emam, K., Melo, W., & Machado, J. C. (2001).The prediction of faulty classes using object-oriented design metrics. Journal of Systems and Software, 56(1), 63-75.

[4] Zhao, M., Wohlin, C., Ohlsson, N., &Xie, M. (1998). A comparison between software design and code metrics for the prediction of software fault content. Information and Software Technology, 40(14), 801-809.

[5] Tomaszewski, P., Lundberg, L., &Grahn, H. (2005). The accuracy of early fault prediction in modified code.In Proceedings of the Fifth Conference on Software Engineering Research and Practice in Sweden (SERPS) (pp. 57-63).

[6] Gyimothy, T., Ferenc, R., &Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. Software Engineering, IEEE Transactions on, 31(10), 897-910.

[7] Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A general software defect-proneness prediction framework. Software Engineering, IEEE Transactions on, 37(3), 356-370.

[8] Koru, A. G., & Liu, H. (2005). Building effective defect-prediction models in practice. Software, IEEE, 22(6), 23-29.

[9] Okutan, A., &Yıldız, O. T. (2014). Software defect prediction using Bayesian networks. Empirical Software Engineering, 19(1), 154-181.

[10] Jureczko, M., &Madeyski, L. (2010, September). Towards identifying software project clusters with regard to defect prediction. In Proceedings of the 6th International Conference on Predictive Models in Software Engineering (p. 9).ACM.

[11] Li, M., Zhang, H., Wu, R., & Zhou, Z. H. (2012). Sample-based software defect prediction with active and semi-supervised learning.Automated Software Engineering, 19(2), 201-230.

[12] Bibi, S., Tsoumakas, G., Stamelos, I., &Vlahavas, I. P. (2006, March). Software Defect Prediction Using Regression via Classification.In AICCSA (pp. 330-336).

[13] Fenton, N., Krause, P., & Neil, M. (2001). A probabilistic model for software defect prediction.IEEE Trans Software Eng.

[14] Challagulla, V. U. B., Bastani, F. B., Yen, I. L., & Paul, R. A. (2008). Empirical assessment of machine learning based software defect prediction techniques. International Journal on Artificial Intelligence Tools, 17(02), 389-400.

[15] Ratzinger, J., Sigmund, T., & Gall, H. C. (2008, May). On the relation of refactorings and software defect prediction. In Proceedings of the 2008 international working conference on Mining software repositories (pp. 35-38). ACM.

[16] Chidamber, S. R., &Kemerer, C. F. (1994). A metrics suite for object oriented design.Software Engineering, IEEE Transactions on,20(6), 476-493.

[17] Di Martino, S., Ferrucci, F., Gravino, C., &Sarro, F. (2011). A genetic algorithm to configure support vector machines for predicting fault-prone components. In Product-Focused Software Process Improvement (pp. 247-261).Springer Berlin Heidelberg.