Component Consistency during Design: A Rule based Evaluation Technique

N.RAJASEKHAR REDDY K.PRAVEENA

Abstract—: The automated detection of faulty modules contained by software systems might lead to reduced development expenses and additional reliable software. In this effort design and development metrics has been used as features to predict defects in given software module using SVM classifier. A rigorous progression of pre-processing steps were applied to the data preceding to categorization, including the complementary of in cooperation classes (faulty or otherwise) and the elimination of a large numeral of repeating instances. The Support Vector Machine in this trial yields a standard accuracy that miles ahead over existing defect prediction models on previously unseen data.

Index Terms—*: Support Vector Machines, Defect prediction, metrics, bug forecasting, Data preprocessing, order effects, training set*

I. INTRODUCTION

SOFTWARE imperfection prediction is the procedure of locating faulty modules in software and is currently an extremely active region of research within the software production community. This is comprehensible as "Faulty software costs businesses $78 billion per year" ([1], published in 2001), consequently any attempt to decrease the number of latent defects that remain inside a deployed arrangement is a worthwhile Endeavour.

Thus the aspire of this revise is to examine the categorization performance of the Support Vector Machine (SVM) for imperfection prediction in the circumstance of eleven data sets from the NASA Metrics Data Program (MDP) warehouse; a collection of data sets generated from NASA software systems and proposed for imperfection prediction investigate. Although imperfection prediction studies contain been approved out with these data sets and a variety of classifiers (including an SVM) in the past, this learn is novel in that thorough data purification methods are used explicitly.

The most important purpose of static code metrics (examples of which consist of the numeral of: lines of code, operators (as projected in [2]) and linearly self-governing paths (as proposed in [3]) in a component) is to provide software scheme managers and suggestion toward the superiority of a software system. Although the entity worth of such metrics has been questioned by many authors within the software manufacturing community (see [4], [5], [6]), they still go on with to be used.

Data mining techniques from the field of artificial intelligence now make it probable to forecast software defects; undesired outputs or personal property produced by software, from staticcodemetrics.

Views toward the value of using such metrics for imperfection prediction are as varied within the software manufacturing community as those toward the value of static code metrics. However, the conclusion within this document suggest that such predictors are helpful, as on the information used in this learn they predict defective modules with an standard accuracy that is miles ahead of existing defect prediction models.

II. RELATED WORK

Most statistical and software metric-based models struggle with formulating accurate defect predictions [3].Fenton [3] considers the poor quality of the data as a major cause for the problem. One approach to address this problem is the application of expert systems and machine learners [1, 2, 3, 7, 11, and 14] in formulating a predictor. Such an approach is plausible since expert systems and machine learners handle noisy data and uncertainty rather well. After considering various reasons for the disappointing results predicting software defects, Fenton [3] concludes that a Bayesian Belief Network would be a possible solution to the problem. Fenton built a tool based on this paradigm called AID, (Assess, Improve, Decide). AID was tested on 28 projects from the Philips Software Centre. The results of this study were promising, however proper training requires great deal of data [4].The application of Neural Networks to the problem of defect detection has received a great deal of attention. Neural Networks have successfully been applied to predict defects in a chemical processing plant. The results were 10 to 20 times better than the application of traditional methods [12]. Hochmann extends the use of Neural Networks to software defects [7].

Thirty classification models were built with an equal distribution of fault-prone and non fault-prone software modules. In the first study, a Genetic Algorithm develops optimal back-propagation networks to detect software defects [7]. In the second study, an Evolutionary Neural Network, (ENN), are compared to Discriminate Analysis. The error rates for Discriminate Analysis were much higher than for

- Associate professor, Department of CSE, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh, India. Rajsekhar007@gmail.com
- Research Scholar, Department of CSE, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh, India. praveenabhaskar12@gmail.com

ENNs. A z-test supported the statistical significance of these findings [7].Evett et al. [1] apply Genetic Programs against several industrial-level repositories in predicting software faults. Their operator set includes add, subtract, multiply, divide, sine, cosine, exponentiation, and logarithms. The operands consist of various product metrics including Halstead's vocabulary, McCabe's cyclomatic complexity, and Lines of Code (LOC). The author's assess their GP models by ranking data sets according to defect counts and comparing the top n percent of actual and predicted models where n ranges from 75 to 90 percent.

III. DATA PREPROCESSIGNAND MACHINE LEARNING

Recognizing the problems with Neural Networks, when applied to software defects and the multi co linearity of the data involved, Neumann [11] developed an enhanced technique for risk categorization called PCA-ANN. This approach combines statistics, pattern recognition and Neural Networks. PCA-ANN essentially enhances the Neural Network by working with the input data to be orthogonal and normalizes the data. The enhanced Neural Network performed significantly better than a pure Neural Network [11]. Hochmann also recognizing the importance of data preparation used Principal Component Analysis to prepare the data set [7]. Finally, Mizuno proposes a data equalization method to improve the performance of an Artificial Neural Network in technical analysis of the stock market [10].

A. DATASET

The data for the machine learning experiments originate from a NASA project, which will be referred to as"KC2." KC2 is a collection of C++ programs containing over 3000 "c" functions. The analysis focuses only on those functions created by NASA developers. This means COTS-based metrics are pruned from the data set. After eliminating redundant data, the final tally consists of metrics from 379 "c" functions. The KC2 data set contains twenty-one software product metrics based on the product's size, complexity and vocabulary. The size metrics include total lines of code, executable lines of code, lines of comments, blank lines, number of lines containing both code and comments, and branch count. Another three metrics are based on the product's complexity. These include cyclomatic complexity, essential complexity, and module design complexity. The other twelve metrics are vocabulary metrics. The vocabulary metrics include Halstead length, Alstead volume, Halstead level, Halstead difficulty, Halstead intelligent content, Halstead programming effort, Halstead error estimate, Halstead programming time, number of unique operators, number of unique operands, total operators, and total operands. The KC2 data set also contains the defect count for each module. The majority of the defect values range from 0 to2.

There are 272 instances of zero defects in the modules, 56 instances of one defect, and 25 instances of two defects. Of the remaining 24 module instances, nine have three defects; four have four defects, five have five defects, two have six defects, one has eight defects, one has ten defects, one has eleven defects, and one has thirteen defects. Thus, ninety percent of the defect data is concentrated in 27 percent of the defect value points.

B. THE SUPPORT VECTOR MACHINE

A Support Vector Machine (SVM) is a supervised engine learning algorithm that can be used for together classification and failure [9]. SVMs are known as highest border classifiers as they discover the best separating overexcited plane between two classes. This procedure can also be functional recursively to allow the division of any amount of classes. Only individuals data points that are positioned nearest to this dividing overexcited plane, known as the *support vectors*, are used by the classifier. This enables SVMs to be used successfully with in cooperation large and small data sets.

Although highest margin classifiers are strictly proposed for linear classification, they can also be used successfully for non-linear categorization (such as the case here) via the use of a kernel purpose. A kernel purpose is used to implicitly map the data points into a higher-dimensional feature space, and to take the inner-product in so as to feature gap [10]. The advantage of using a kernel purpose is that the data is more likely to be linearly detachable in the higher feature gap. Additionally, the genuine mapping to the higher-dimensional gap is by no means needed.

There are a digit of different kinds of kernel functions (any nonstop symmetric optimistic semi-definite purpose will suffice) including: linear, polynomial, Gaussian and sigmoidal. Each has varying characteristics and is suitable for different difficulty domains. The one used at this time is the *Gaussian radial foundation function* (RBF), as it can handle non-linear problems, requires smaller amount parameters than other non-linear kernels and is computationally less challenging than the polynomial kernel [11].

When an SVM is used with a Gaussian RBF kernel, there are two user- particular parameters, C and γ. C is the mistake cost parameter; a changeable that determines the trade-off between minimizing the preparation error and maximizing the border. 7 control the width / radius of the Gaussian RBF. The presentation of an SVM is largely dependent on these parameters, and the most favorable values require to be determined *for every training set* via a methodical search.

C. DATA

The data used within this study was obtained from the NASA Metrics Data Program (MDP) depository. This depository currently contains thirteen information sets, each of which represent a NASA software scheme / subsystem and have the static code metrics and corresponding burden data for every comprising unit. Note that a module in this field can refer to a purpose, process or technique. Eleven of these thirteen data sets were used in this learn: brief particulars of each are exposed in Table 1. A total of 42 metrics and a unique unit identifier comprise each data set (see Table 5, located in the appendix), with the exemption of MC1 and PC5 which do not have the decision density metric.

Name	Language	Total KLOC	No. of Modules	Defect Modules Ratio in %
CM1	C	20	505	10
KC3	Java	18	458	9
KC4	Perl	25	125	49
MC1	C & C++	63	9466	0.7
MC2	C	6	161	32
MW1		8	403	8
PC1		40	1107	7
PC2		26	5589	0.4
PC3		40	1563	10
PC4		36	1458	12
PC5	C++	164	17168	3

Table 1: List of datasets used

IV. METHOD OF IMPLEMENTATION

A. Data Pre-processing

The procedure for cleansing each of the data sets used in this study is as follows:

Primary Data Set Modifications every of the data sets primarily had their unit identifier and *error density* characteristic removed, as these are not necessary for categorization. The *error count* characteristic was then transformed into a binary goal characteristic for each instance by assigning all principles greater than zero to defective, non-defective otherwise.

Removing repetitive and incompatible Instances recurring feature vectors, whether with the identical (repeated instances) or dissimilar (inconsistent instances) class labels, are a known difficulty within the data mining community [12].

Ensuring that training and testing sets do not contribute to instances guarantees that all classifiers are being experienced against *previously unnoticed data*. This is very important as testing a analyst upon the data used to instruct it can greatly overestimate presentation [12]. The removal of incoherent items from training data is also significant, as it is clearly illogical in the context of double classification for a classifier to correlate the same data point with equally classes.

Carrying out this pre-processing phase showed that some data sets (namely MC1, PC2 and PC5) had an overwhelmingly elevated number of repeating instances (79%, 75% and 90% respectively, see Table 2). Although no clarification has yet been found for these high numbers of repeated instances, it appears highly improbable that this is a true depiction of the data, i.e. that 90% of modules within a system / subsystem could perhaps have the same number of: lines, comments, operands, operators, exceptional operands, exceptional operators, conditional statements, etc.

Table 2: The result of removing all repeated and inconsistent instances from the data

Name	Original Instances	Instances Removed	Removed ratio in %
CM1	505	51	10
KC3	458	134	29
KC4	125	12	10
MC1	9466	7470	79
MC2	161	5	3
MW1	403	27	7
PC1	1107	158	14

Removing Constant Attributes If an attribute has a preset value throughout all instances then it is apparently of no use to a classifier and must be removed.

Each data set had connecting 1 and 4 attributes removed through this phase with the exception of KC4 that had a total of 26. Particulars are not shown here outstanding to space limitations.

Missing Values Missing values are those that are inadvertently or otherwise missing for a particular attribute in a meticulous instance of a data set. The only missing standards within the data sets used in this study were within the *decision density* attribute of information sets CM1, KC3, MC2, MW1, PC1, PC2, and PC3.

Manual inspection of these misplaced values indicated that they were almost certainly theoretical to be representing zero, and were replaced consequently.

Balancing the Data All the information sets used within this study, with the exception of KC4, enclose a much larger amount of one class (namely, non-defective) than they do the other. When such *excessive* data is used with a supervised classification algorithm such as an SVM, the classifier will be predictable to over predict the majority class [10], as this will make lower error rates in the experiment set.

There are different techniques that can be used to poise data (see [13]). The approach taken here is the simplest still, and involves randomly under sampling the majority class awaiting it becomes equal in size to that of the alternative class. The number of instances that were detached during this under sample process, along with the closing number of instances contained within each information set, are shown in Table 3.

Table 3: The result of balancing each data set

Name	Instances Removed	Removed	Final no. of Instances
CM1	362	80	92
KC3	240	74	84
KC4	1	1	112
MC1	1924	96	72
MC2	54	35	102

MW1	320	85	56
PC1	823	87	126
PC2	1360	97	42
PC3	1133	79	300
PC4	990	74	352
PC5	862	48	942

Normalization All values within the information sets used in this study are numeric, so to avoid attributes with a huge range dominating the categorization model all values were normalized between -1 and +1. Note that this pre-processing phase was performed immediately prior to training for each preparation and testing set, and that each preparation / testing set pair were scaled in the equal manner [11].

Randomizing Instance Order The order of the instances within each information set was randomized to secure against *order effects*, where the performance of a predictor fluctuates outstanding to certain orderings within the data [14].

Experimental Design

When splitting each of the information sets into preparation and testing sets it is important to ameliorate potential anomalous results. To this end we use five-fold cross legalization. Note that to decrease the effects of sampling bias introduced when erratically splitting information set into five bins, the cross-validation progression was repeated 10 times for each information set in each iteration of the experiment (described below).

As mentioned in Section 2.2, an SVM with an RBF kernel requires the selection of optimal principles for parameters C and 7 for maximal presentation. Both values were chosen for each preparation set using a five-fold *grid search* (see [11]), a progression that uses cross-validation and a wide range of potential parameter values in a regular fashion. The pair of values that yield the maximum average accuracy are then taken as the finest parameters and used when generating the closing model for classification.

Due to the elevated percentage of information lost when harmonizing each information set (with the exception of KC4), the experiment is recurring fifty times. This is in order to extra minimize the effects of sampling bias introduced by the chance under sampling that takes place during complementary.

Pseudo code for the complete experiment carried out in this revise is shown in Fig 3. Our chosen SVM situation is LIBSVM [15], an open basis library for SVM experimentation.

Assessing Performance

The measure used to evaluate predictor performance in this revise is *accuracy*. Accuracy is distinct as the ratio of instances correctly classified out of the whole number of instances. Although simple, accuracy is a suitable performance measure for this learns as each test set is fair. For imbalanced test sets additional complicated measures are necessary.

Results

The average consequences for each data set are shown in Table 4. The results demonstrate an average accuracy of 70% across all 11 information sets, with a range of 64% to 82%. Note that there is a moderately high deviation shown within the consequences. This is to be expected due to the huge amount of information lost during balancing and supports the result for the experiment being repeated fifty times (see Fig. 1). It is prominent that the accuracy for some data sets is tremendously high, for example with statistics set PC4, four out of every five modules were being properly classified.

The consequences show that all statistics sets with the exception of PC2 have a mean accurateness greater than two typical deviations away from 50%. This shows the statistical significance of the classification consequences when compared to a dumb classifier that predicts all one class (and therefore scores an accuracy of 50%). In fig 2 it is clearly evident that SVM based defect prediction model is miles ahead in performance over GA based defect prediction.

Table 4: The results obtained from this study.

Name	Accuracy Mean in %	Std.
CM1	68	5.57
KC3	66	6.56
KC4	71	4.93
MC1	65	6.74
MC2	64	5.68
MW1	71	7.3
PC1	71	5.15
PC2	64	9.17
PC3	76	2.15
PC4	82	2.11
PC5	69	1.41
Total	70	5.16

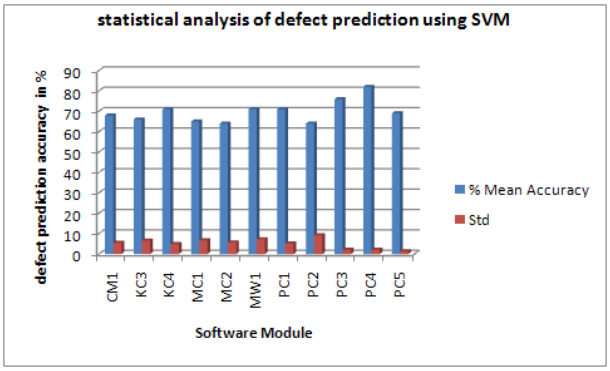

Fig 1: statistical analysis of defect prediction accuracy

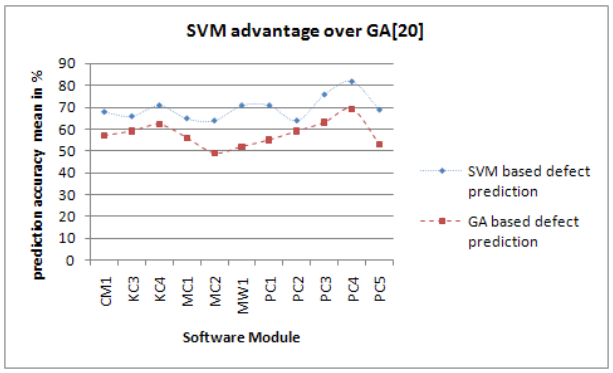

Fig 2: Performance Analysis of defect detection using SVM over GA

ANALYSIS:

Earlier studies ([16], [17], [18]) have also used information from the NASA MDP repository and an SVM classifier. Some of these studies mentioned about information pre-processing, however we consider that it is important to clearly carry out all of the information cleansing stages described here. This is particularly true with regard to the elimination of repeating instances, ensuring that all classifiers are being experienced against previously unseen information.

The elevated number of repeating instances originate within the MDP information sets was surprising. Brief analysis of other fault prediction information sets showed a repeating average of just 1.4%. We are consequently suspicious of the suitability of the information held within the MDP repository for fault prediction and believe that earlier studies which have used this information and not carried out appropriate information cleansing methods may be reporting inflated presentation values.

An example of such a revise is [18], where the authors use an SVM and four of the NASA information sets, three of which were used in this learn (namely CM1, PC1 and KC3). The authors make no mention of information pre-processing other than the use of quality selection algorithm. They then go on to account a minimum average *precision*, the relation of correctly predicted defective modules to the whole number of imperfect modules, of 84.95% and a minimum standard recall, the ratio of defective modules detected as such, of 99.4%. We believe that such elevated classification rates are highly unlikely in this difficulty domain due to the confines of static code metrics and that not carrying out suitable data cleansing methods may have been a issue in these high consequences.

CONCLUSION

This study has shown that on the information studied here the carry Vector Machine can be used effectively as a classification method for fault prediction. We hope to improve upon these consequences in the near future however via the use of a one- class SVM; an addition to the original SVM algorithm that trains upon only imperfect examples, or a more sophisticated balancing method such as SMOTE (Synthetic Minority Over-sampling method).

Our consequences also show that earlier studies which have used the NASA information may have exaggerated the extrapolative power of static code metrics. If this is not the casing then we would recommend the explicit certification of what data pre-processing methods have been applied. Static regulations metrics can only be used as probabilistic statements toward the excellence of a module and further study may need to be undertaken to describe a new set of metrics particularly designed for defect prediction.

The significance of information analysis and information quality has been highlighted in this learn; especially with regard to the elevated quantity of repeated instances establish within a number of the information sets. The issue of information quality is extremely important within any information mining experiment as deprived quality data can threaten the validity of together the consequences and the conclusions strained from them [19].

FUTURE DIRECTIONS

The process of equalizing data in an implicitly starved environment can be considered to be valuable when applied to a machine learning in the defect prediction domain. Other research has found it useful when applied to neural networks in the financial forecasting domain [10].The amount of work done in this area remains limited to a few studies. It seems logical to assume that this methodology could be equally useful when applied to other machine learners, as well as other domains. More research done in this area would help validate the conclusions drawn in this paper. Another possible area of research involves the effects of data equalization on the performance of the machine learner. It would be useful to know in which situations this methodology is appropriate and when it is infeasible due to the size of the data and its effects upon performance. Finally, it would be useful to apply to the results from one NASA defect data set to other NASA defect data sets to determine if the solution is transferable. Also, a comparison can be made with the solution obtained from the original data set and from the equalized data set when applied to a different data set.

REFERENCES

[1]. Levinson, M.: Lets stop wasting $78 billion per year. CIO Magazine (2001)

[2]. Halstead, M.H.: Elements of Software Science (Operating and programming sys¬tems series). Elsevier Science Inc., New York, NY, USA (1977)

[3]. McCabe, T.J.: A complexity measure. In: ICSE '76: Proceedings of the 2nd international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1976) 407

[4]. Hamer, P.G., Frewin, G.D.: M.H. Halstead's Software Science - a critical exami¬nation. In: ICSE '82: Proceedings of the 6th international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1982) 197¬206

[5]. Shen, V.Y., Conte, S.D., Dunsmore, H.E.: Software Science Revisited: A critical analysis of the theory

and its empirical support. IEEE Trans. Softw. Eng. 9(2) (1983) 155-165

[6]. Shepperd, M.: A critique of cyclomatic complexity as a software metric. Softw. Eng. J. 3(2) (1988) 30-36

[7]. Sommerville, I.: Software Engineering: (8th Edition) (International Computer Sci¬ence Series). Addison Wesley (2006)

[8]. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. Software Engineering, IEEE Transactions on 33(1) (Jan. 2007) 2-13

[9]. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning). The MIT Press (2001)

[10]. Sun, Y., Robinson, M., Adams, R., Boekhorst, R.T., Rust, A.G., Davey, N.: Using sampling methods to improve binding site predictions. In: Proceedings of ESANN. (2006)

[11]. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classifica¬tion. Technical report, Taipei (2003)

[12]. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Tech¬niques. Second edn. Morgan Kaufmann Series in Data Management Systems. Mor¬gan Kaufmann (June 2005)

[13]. Wu, G., Chang, E.Y.: Class-boundary alignment for imbalanced dataset learning. In: ICML 2003 Workshop on Learning from Imbalanced Data Sets. (2003) 49-56

[14]. Fisher, D.: Ordering effects in incremental learning. In: Proc. of the 1993 AAAI Spring Symposium on Training Issues in Incremental Learning, Stanford, California (1993) 34-41

[15]. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[16]. Li, Z., Reformat, M.: A practical method for the software fault-prediction. Infor¬mation Reuse and Integration, 2007. IRI 2007. IEEE International Conference on (Aug. 2007) 659-666

[17]. Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: A proposed framework and novel findings. Software Engineering, IEEE Transactions on 34(4) (2008) 485-496

[18]. Elish, K.O., Elish, M.O.: Predicting defect-prone software modules using support vector machines. J. Syst. Softw. 81(5) (2008) 649-660

[19]. Liebchen, G.A., Shepperd, M.: Data sets and data quality in software engineering. In: PROMISE '08: Proceedings of the 4th international workshop on Predictor models in software engineering, New York, NY, USA, ACM (2008) 39-44.

[20]. Kaminsky, K. and G. Boetticher. "How to Predict More with Less, Defect Prediction Using Machine Learners in an Implicitly Data Starved Domain," 8th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, FL, July 18-21, 2004.

AUTHOR'S DESCRIPTION

N.Rajasekhar Reddy was born in Madanapalli, February 28.He was received Bachelor's degree in Computer Science in S.V University and M.Tech degree from Satyabama University respectively. After working as a research scholar and an Associate professor in the Dept. of Computer Science and Engineering, Madanapalle Institute of Technology and Science, Andhra Pradesh, India. His research interest includes Software Engineering, Software Quality Assurance and Testing. He was published 8 international journal papers and 5 National journal papers in Software Engineering. He is a member of SCIE, ISTE, and IEEE, SEI.

K.Praveena was received Bachelor's degree in Computer Science in S.V University and pursuing M.Tech (CSE) Studying from J.N.T University Anatapur respectively. After working as a research assistant Computer Science and Engineering Dept Madanapalle. And an Assistant professor in the Dept. of Computer Science and Andhra Pradesh, India. His research interest includes Software Engineering, Software Quality.She is pursuing her M.Tech project work under the esteemed guidance of N.Rajasekhar Reddy.