ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, **UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019**

An analysis of feature selection methods for predicting software defects

E. Sreedevi [1],

[1] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P. – 522302.

PremaLatha V [2]

[2] Department of Computer Science and Engineering Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P. – 522302.

Abstract:

Building software that is free from errors is essential as software has become an essential component of human life. To plan an exceptional maintenance approach, software modules that are prone to defects must be identified before the software project is deployed. Early detection of software modules that are prone to defects can help create an appropriate process improvement plan in a time and cost-justified manner. This may also result in better product releases and high levels of client satisfaction down the road. Since defect prediction and measurement are indirect and dependent on multiple factors, they are critical issues in software development. Consequently, it would be more beneficial to choose a reasonable set of qualities that are pertinent and significant for defect prediction in any software module rather than taking into account all the attributes. Techniques for feature extraction and selection are used to identify the set of important attributes that are definitely influencing software module fault prediction Reducing the amount of characteristics following feature selection can improve the efficiency of prediction models, which can then be applied to identify faulty modules in a given set of inputs. The performance evaluation of several methods for feature extraction and selection is eventually drafted in this study. Four datasets from the PROMISE software engineering repository have been used in the experiment. The investigation and outcome indicate that using feature selection could increase accuracy.

KeyWords: Software Defect prediction, Software metrics, Feature selection, Accuracy, NASA datasets.

1. Introduction

As Lord Kelvin [1] put it, "If you can't measure it. You are unable to handle it. When data is not adequately available, data management becomes crucial. Based on software metrics, software defect prediction models are typically used to identify faulty software modules. Before software is tested and used, data is gathered and used for training these modules. All of the software's fault-prone modules could be identified by the classification model. Generally speaking, a learning algorithm rarely succeeds in creating a model that is flawless. Therefore, in order to maximise the model's prediction accuracy and create the best model possible, academics and practitioners must work hard. Previous research has unequivocally demonstrated that removing superfluous and irrelevant variables from the original data set enhances the performance of these models. There are two methods for reducing features: feature extraction and feature selection. In each of these scenarios, we either preserve or improve classification accuracy while reducing classifier complexity. This study examined a number of feature selection methods using the NASA Metric Data Programme as a foundation [2]. Predicting software defects is the primary purpose for these data sets. There are thirteen data sets in the NASA data set, and each data set includes defective modules. Only eight of these thirteen datasets—JM1, KC1, MC2, MW1, PC1, PC3, PC4, and PC5—were used. Several feature selection strategies, such as Chi-Chi-squared (CS), Gain ratio (GR), and ReliefF (RF), were applied to these eight data sets. We employed J48 Decision tree classification algorithm for performance evaluation on NASA [2] data sets using WEKA tool [3]. In addition to being utilised for performance evaluation of the aforementioned algorithm, this tool is a library of several machine learning algorithms [4].

Use of performance assessors such as recall, precision, F-measure, ROC, TPRate, and FPRate is done when feature selection algorithms are applied to datasets. The dataset utilised in this work comes from NASA MDP [2], processing is done using the Weka tool [3], and conclusions are based on specific result evaluators. The structure of this document is as follows: In Section 2, the key findings from related research in this field are reviewed. The explanation of the dataset used in our investigation is provided in Section 3. Section 4 provides an explanation of different feature selection methods. Comparative examination of the several feature selection methods in Tables 3 and 4 is discussed in Section 5, and section 6 concludes with findings.

2. Related work

A crucial pre-processing stage in machine learning, feature selection has found widespread application in a variety of fields, including text classification [5], bioinformatics, and biological networks [6]. It is well knowledge that several metrics may be correlated with software system defect-proneness. Consequently, improving classification performance may require deleting uncorrelated measures [7]. Using sixteen software data sets, Khoshgoftaar et al. [8] investigated seven filter-based features ranking approaches for comparison. Khoshgoftaar et al. investigated a technique that uses the signal-to-noise ratio, which is not commonly used as a software metric. Across all examined data sets, the Information Gain and signal to noise ratio had the best average defect classification performance [7].

For defect prediction, Sonali Agrwal and Divya Tomar [9] suggested an LSTSVM Model based on feature selection. Their study reveals that the suggested model may be helpful in forecasting software quality and discloses the efficacy of the suggested feature selection based LSTSVM strategy in predicting faulty software modules [9]. According to Graves et al. [10], a file's previous change count is a reliable indicator of faults. In particular, they discovered that when the frequency of module changes is considered, the amount of LOC of a module is not useful in forecasting failures. The authors came to the conclusion that the size and structural metrics could not provide as much valuable information as the change history [11].

3. NASA Data Sets

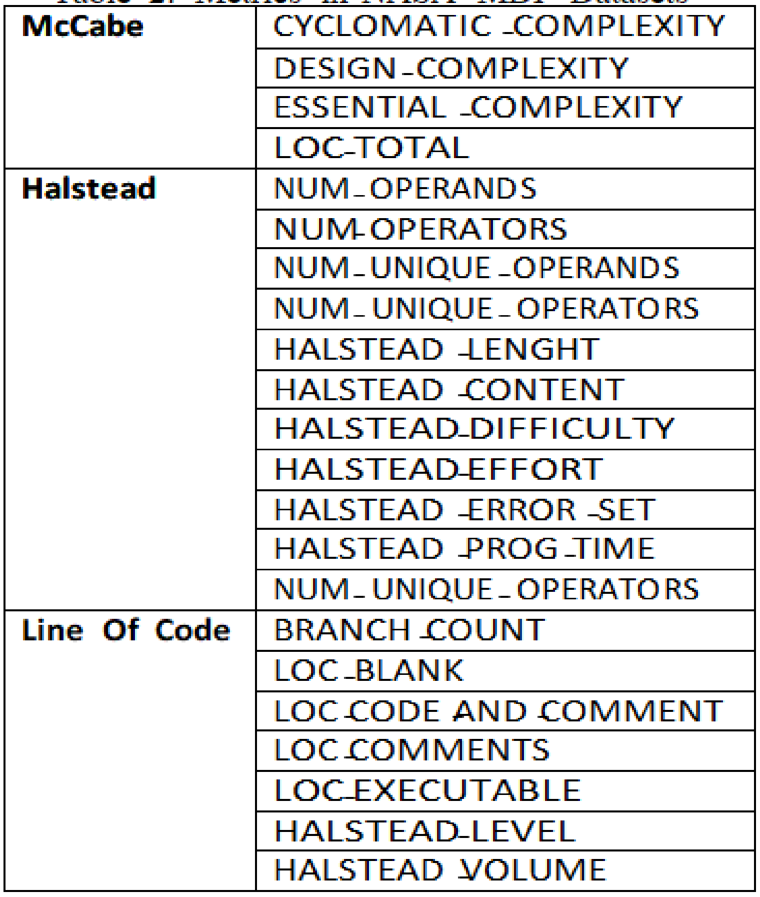
The NASA IV&V MDP Metric Data Repository makes the datasets utilised in our investigation available to the general public [2]. The software engineering research community has made extensive use of the NASA MDP [4]. Eight datasets—JM1,KC1,MC2, MW1,PC1,PC3, PC4, and PC5—are used in this study. Table 1 lists a few metrics that are present in these datasets.

Name	Language	No. Instance	No. Attributes
JM1	C	9593	22
KC1	C++	2096	22
MC2	C	127	40
MW1	C	264	38
PC1	C	759	38
PC3	C	1125	38
PC4	C	1399	38
PC5	C++	17001	39

Table 1: NASA MDP data sets.

These datasets include some real-time projects that are made up of metrics such as LOC metrics (lines of code, number of unique operators, number of operands, LOC Blank, LOC Comment, Branch Count, etc.), Halstead metrics (Halstead Length, Halstead Difficulty, Halstead Effort, Halstead Level, Halstead Error Estimate, Halstead Programming Time, and Halstead Volume, etc.), and McCabe metrics (Cyclomatic Complexity Metrics, Cyclomatic Density Metric, Normalised Cyclomatic Complexity Metrics, etc.). Table 2 provides the specifics of the metrics found in the datasets.

Table 2: Metrics in NASA MDP Datasets



McCabe	CYCLOMATIC_COMPLEXITY
	DESIGN_COMPLEXITY
	ESSENTIAL_COMPLEXITY
	LOC_TOTAL
Halstead	NUM_OPERANDS
	NUM_OPERATORS
	NUM_UNIQUE_OPERANDS
	NUM_UNIQUE_OPERATORS
	HALSTEAD_LENGHT
	HALSTEAD_CONTENT
	HALSTEAD_DIFFICULTY
	HALSTEAD_EFFORT
	HALSTEAD_ERROR_SET
	HALSTEAD_PROG_TIME
	NUM_UNIQUE_OPERATORS
Line Of Code	BRANCH_COUNT
	LOC_BLANK
	LOC_CODE AND_COMMENT
	LOC_COMMENTS
	LOC_EXECUTABLE
	HALSTEAD_LEVEL
	HALSTEAD_VOLUME

4. Research Methodology

Better solutions and increased discriminating (or classification) power are implied by features, which define the classification function and provide information about the target. This might not always be true. A better solution does not always imply more features. When the training set is not overly large and the number of training instances is fixed, it usually happens that the classifier's performance increases initially and subsequently declines as the number of features is increased. The explanation is that certain traits may not matter. Because we are attempting to determine which instances are close together, these irrelevant variables produce noise in algorithms like K-nearest neighbour and confuse the learning algorithm, resulting in incorrect results. We might be using duplicate features. The performance of the learning algorithm may deteriorate if we have a fixed number of training samples and redundant features that don't bring any new information. Particularly when there aren't enough training examples or computing resources, these superfluous and irrelevant aspects can be confusing to learners. Depending on the algorithm we select, searching may take longer when there are more features in the search space and hypothesis space. Furthermore, because there aren't enough training instances, we can't work with more features because it would create overfitting. Therefore, these factors play a part in the phenomenon known as the curse of dimensionality. An excessive number of features or dimensions can cause the learning method to deteriorate and need more computational time. We must perform feature reduction in order to escape the dimensionality curse. Feature selection and feature extraction are the two categories of feature reduction techniques. In each of these scenarios, we simplify the complexity of the classifier and either increase or retain classification accuracy.

4.1 Feature Selection

We wish to discover subset F 1, (F 1 ⊂ F) = {x1, x1,....x1 } in feature selection given initial set of features F = {x1, x2, x3,...xN } given N number of features. How is a subset chosen? There are 2N different subsets that can be constructed from a given collection of N features. We can count each of these subsets and evaluate their quality. because there are exponentially many subsets. Thus, we require a technique that operates in a fair amount of time. The best, heuristic, and randomised approaches are among those we can employ for feature selection. We can assess subsets using supervised and unsupervised techniques. We don't evaluate the subset over the training instances in unsupervised approaches. We use what are known as filter methods to assess the information content in an unsupervised manner. These techniques, known as wrap-per approaches, are used in supervised algorithms to evaluate feature subsets utilising learning algorithms. In filter techniques, the search algorithm generates a feature subset based on the search algorithm approach, which is then utilised to determine the final feature subset after it has been reviewed for information content.

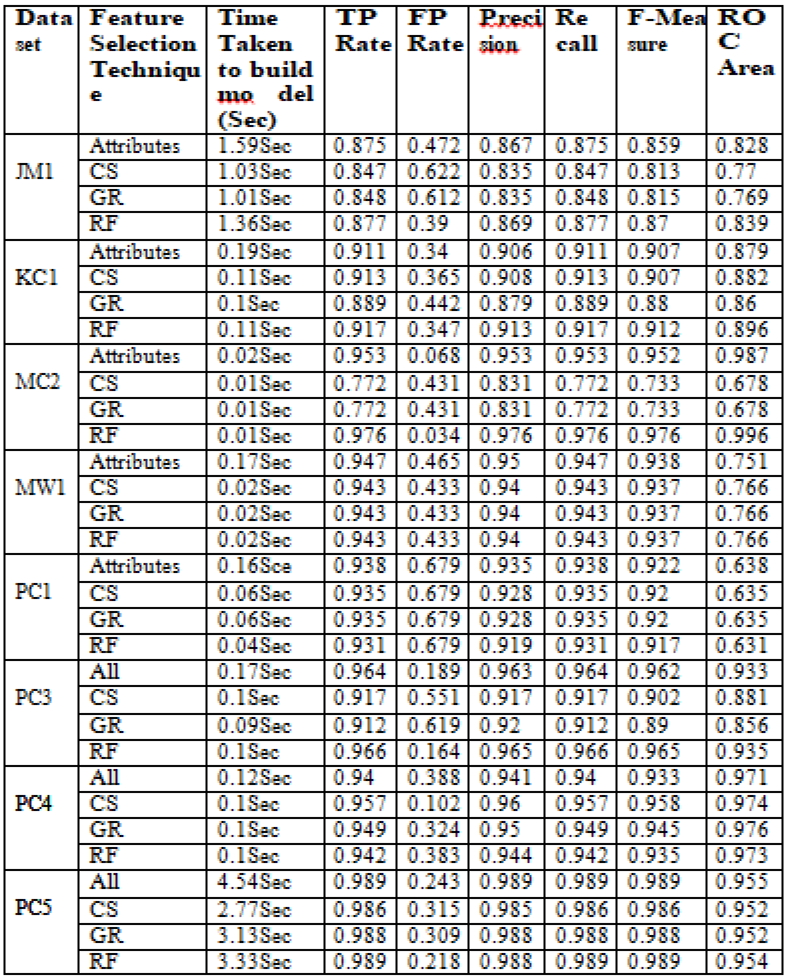
After the entire module is finished, we will have a feature subset that is used by a machine learning algorithm. In the wrapper method, the search algorithm outputs the feature subset that is again used with a machine learning algorithm. The prediction accuracy obtained is fed to the search algorithm. There are two types of feature selection techniques: univariate and multivariate. Uinvariate examines every attribute separately from the others. Multivariate takes into account every aspect at once.

Each feature in the feature selection technique is assigned a score, or rating, which enables the optimal combination of features to be chosen. We employed popular filter-based feature selection methods in this work, including Chi-Square (CS), Gain Ratio (GR), and RelifF (RF).The distribution of the class in relation to the values of the provided feature is examined using the Chi-square, $\chi2$ test [12]. The null hypothesis states that there is no association, meaning that there is an equal chance of each value occurring in every class [12]. Gain ratio that penalises characteristics with multiple values. It is a method based on probability. An variant of the RelifF algorithm that can manage noise and multiclass datasets is RelifF, an instance-based feature ranking method [12]. The J48 Decision tree classification technique, which takes the shape of a tree structure with each node representing either a decision or a leaf node, was employed in our investigation. Referred to as classification trees, these trees are employed to forecast a case's membership in a categorical dependent variable class based on the measurement of one or more predictor variables [4].

5. Experimental Setup

We made use of eight MDP datasets, where the variable defect indicates whether or not a module has been determined to be fault-prone. Performance evaluators such as recall, precision, F-measure, ROC, TPRate, and FPRate are used to analyse the results. The comparison of outcomes with all characteristics and selected features using various feature selection methods is shown in Table 3.

Table 3: Results of evaluators with different feature selection techniques



Data set	Feature Selection Technique	Time Taken to build mo del (Sec)	TP Rate	FP Rate	Precision	Re call	F-Mea sure	ROC Area
JM1	Attributes	1.59Sec	0.875	0.472	0.867	0.875	0.859	0.828
	CS	1.03Sec	0.847	0.622	0.835	0.847	0.813	0.77
	GR	1.01Sec	0.848	0.612	0.835	0.848	0.815	0.769
	RF	1.36Sec	0.877	0.39	0.869	0.877	0.87	0.839
KC1	Attributes	0.19Sec	0.911	0.34	0.906	0.911	0.907	0.879
	CS	0.11Sec	0.913	0.365	0.908	0.913	0.907	0.882
	GR	0.1Sec	0.889	0.442	0.879	0.889	0.88	0.86
	RF	0.11Sec	0.917	0.347	0.913	0.917	0.912	0.896
MC2	Attributes	0.02Sec	0.953	0.068	0.953	0.953	0.952	0.987
	CS	0.01Sec	0.772	0.431	0.831	0.772	0.733	0.678
	GR	0.01Sec	0.772	0.431	0.831	0.772	0.733	0.678
	RF	0.01Sec	0.976	0.034	0.976	0.976	0.976	0.996
MW1	Attributes	0.17Sec	0.947	0.465	0.95	0.947	0.938	0.751
	CS	0.02Sec	0.943	0.433	0.94	0.943	0.937	0.766
	GR	0.02Sec	0.943	0.433	0.94	0.943	0.937	0.766
	RF	0.02Sec	0.943	0.433	0.94	0.943	0.937	0.766
PC1	Attributes	0.16Sce	0.938	0.679	0.935	0.938	0.922	0.638
	CS	0.06Sec	0.935	0.679	0.928	0.935	0.92	0.635
	GR	0.06Sec	0.935	0.679	0.928	0.935	0.92	0.635
	RF	0.04Sec	0.931	0.679	0.919	0.931	0.917	0.631
PC3	All	0.17Sec	0.964	0.189	0.963	0.964	0.962	0.933
	CS	0.1Sec	0.917	0.551	0.917	0.917	0.902	0.881
	GR	0.09Sec	0.912	0.619	0.92	0.912	0.89	0.856
	RF	0.1Sec	0.966	0.164	0.965	0.966	0.965	0.935
PC4	All	0.12Sec	0.94	0.388	0.941	0.94	0.933	0.971
	CS	0.1Sec	0.957	0.102	0.96	0.957	0.958	0.974
	GR	0.1Sec	0.949	0.324	0.95	0.949	0.945	0.976
	RF	0.1Sec	0.942	0.383	0.944	0.942	0.935	0.973
PC5	All	4.54Sec	0.989	0.243	0.989	0.989	0.989	0.955
	CS	2.77Sec	0.986	0.315	0.985	0.986	0.986	0.952
	GR	3.13Sec	0.988	0.309	0.988	0.988	0.988	0.952
	RF	3.33Sec	0.989	0.218	0.988	0.989	0.989	0.954

The accuracy comparison with respect to true positive rate for both selected and all attributes is presented in Table 4.

Table 4: The accuracy comparison with all attributes and selectedattributes

Data set Selection Technique	JM1	KC1	MC2	MW1	PC1	PC3	PC4	PC5
Attributes	0.874	0.911	0.952	0.946	0.938	0.963	0.939	0.989
CS	0.846	0.913	0.771	0.943	0.935	0.917	0.957	0.986
GR	0.847	0.889	0.771	0.943	0.935	0.912	0.949	0.985
RF	0.877	0.917	0.976	0.943	0.931	0.966	0.942	0.988

Figure 1 describes the accuracy comparison in terms of true positive rate with all attributes and selected attributes.

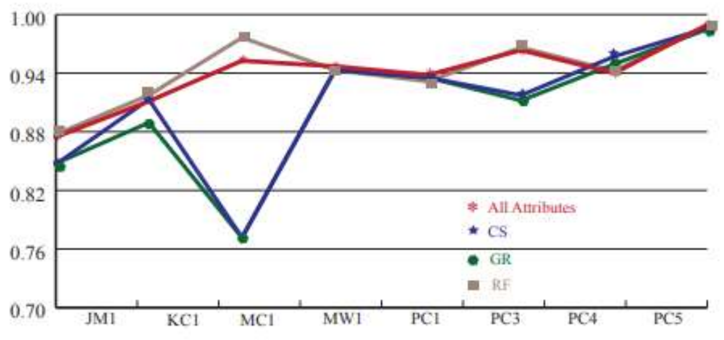


Figure 1: Accuracy Comparison

6. Conclusion

The usefulness of various feature selection strategies for reducing the number of dimensions in a dataset and raising classifier accuracy has been assessed. Accurate prediction models are created with the aid of feature selection techniques. Using a variety of feature selection techniques on a dataset and comparing the outcomes will help determine which performance view is optimal. This will help identify the features that will best reveal the problem's structure to a general learning algorithm.

7. Reference

[1] IssamH.laradji, Mohammad Alshyeb, LahouariGhouti, "Software defect prediction using ensemble learning on selected features", Inform.Sfotw.Technol, 2014

[2] Abdullateef O. Balogun, Amos O. Bajeh,Victor A. Orie and Ayisat W. YusufAsaju, "Software Defect Prediction Using Ensemble Learning: An ANP Based Evaluation Method",FUOYE Journal of Engineering and Technology, Vol. 3, No. 2, September 2018.

[3] Mohammad Zubair Khan, "Hybrid Ensemble Learning Technique for Software Defect Prediction", I. J. Modern Education and Computer Science, 2017.

[4] Abdullah Alsaeedi, Mohammad Zubair Khan, "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study", Journal of Software Engineering and Applications, Vol. 12, 2019, pp. 85-100 .

[5] Prasanth Y., Sreedevi E., Gayathri N. and Rahul A.S., "Analysis and implementation of ensemble feature selection to improve accuracy of software defect detection

model", Joiurnal of advanced research in Dynamical and Control Systems, Vol 9, No. 18, 2017, pp.601-613

[6] E. Chandra Blessie, E. Karthikeyan, "Sigmis: A Feature Selection Algorithm Using Correlation Based Method", Journal of Algorithms & Computational Technology Vol. 6, No. 3,2012

[7] Ran Li, Lijun Zhou, Shudong Zhang, Hui Liu,Xianhgyang Huang, Zhong Sun, "Software defect prediction based on Ensemble learning", Proceedings of the 2019 2nd International Conference on Data Science and Information Technology, July 1- 6, 2019

[8] Sivakumar. S, SoumyaRanjanNayak, Ashok Kumar, S. Vidyanandini, GopinathPalai, "An empirical study of supervised learning methods for Breast Cancer Diseases", International Journal for Light and Electron Optics , Vol. 175, 2018, pp. 105-114

[9] Sivakumar S, Sreedevi E, Premalatha V, Haritha D, "Parallel Defect Detection Model on Uncertain Data for GPUs Computing by a Novel Ensemble Learning", Applications of Artificial Intelligence for Smart Technology,2018, pp. 146-163

[10] ShaikRazia, "A Comparative study of machine learning algorithms on thyroid disease prediction", International Journal of Engineering and Technology (UAE), Vol. 7(2.8), 2008, pp. 315-319

[11] Sivakumar. S, Ganesan. P,andSundar. S. "A MMDBM Classifier with CPU and CUDA GPU computing in various sorting procedures", International Arab Journal of Information Technology, Vol. 14, No.7, 2017,pp. 897-906

[12] XiaotaoRong, Feixiang Li, Zhihua Cui, "A model for software defect prediction using support vector machine based on CBA", Int. J. Intelligent Systems Technologies and Applications, Vol. 15, No. 1, 2016

[13] Rajesh Kumar. E and K.V.S.N. Rama Rao. "Suicide Prediction in Twitter Data using Mining Techniques: A Survey", International Conference on Intelligent Sustainable Systems (ICISS 2019), 2020,pp.122-131

[14] SaiqaAleem , Luiz Fernando Capretz andFaheem Ahmed, "Benchmarking machine learning techniques for software defect detection", International Journal of Software Engineering & Applications (IJSEA), Vol.6, No.3, May 2015

[15] Yan Naungsoe, Paulus InsapSantosa, Rudy Hartanto, "Software defect prediction using Random Forest algorithm", 12th South East Asian Technical University Consortium (SEATUC),2018

[16] Sreedevi, E, PremaLatha V, Sivakumar. S. "A Comparative Study on New Classification Algorithm using NASA MDP Datasets for Software Defect Detection", 2 ndInternational Conference on Intelligent Sustainable Systems (ICISS 2019), 2019, pp.312-317.