Investigation of Software Defects Prediction Based on Classifiers (NB, SVM, KNN and Decision Tree)

Amjad Hudaib, Fawaz AL zaghoul, Jaber AL Widian

IS Department, Jordan University, Jordan
j.alwedyan@arabou.edu.sa

Abstract: Constructing predictive model process can be considered as one important approach to improve software quality and testing efficiency. Testing and maintenance phases are the main factors which have to be taken when the cost estimation is carried out for the software product. Thus, accuracy of defects prediction will reduce the efforts in the testing process and give estimation for the product's required maintenance. This paper main goal is to investigate the potential use of automated data mining techniques in software defect problem. The results of this paper showed that the performance of the compared algorithms have a potential results against the software defects problem. Moreover, Naïve Bayesian method (NB), Support Vector Machine (SVM), Decision Trees, and K-nearest Neighbor (KNN) have been investigated on NASA data set. The bases of our comparison are the most popular evaluation measures for the classification techniques (F1, Precision, and Recall). The average of the three measures obtained against false data set indicated that the NB classifier outperformed the SVM, KNN and Decision Tree algorithms.

[Amjad Hudaib, Fawaz AL zaghoul, Jaber AL Widian. **Investigation of *Software* Defects Prediction *Based on Classifiers (NB, SVM, KNN and Decision Tree)*.** *J Am Sci* 2013;9(12):381-386]. (ISSN: 1545-1003). http://www.jofamericanscience.org. 52

Keyword: Software defect; data mining; classification

1. Introduction

As any other product built by humans software is subjected to defect, it is never the developer intention to build defected software but it is the nature of defects to creep into software. Predicting which parts of software are likely to be defected can reduce effort. Developers store large amounts of information about software and its attributes, such information can be used to build useful understanding and prediction of defects.

Testing is the process of executing programs with the intention of finding defects. A software defect is an error, flaw, mistake, failure, or fault in a computer program or system that produces incorrect or unexpected results, or causes it to behave in unintended way (Kaur and Pallavi, 2013). While keeping in mind the required quality level, software developers are always in pursuit of ways to reduce cost and time, one way of doing so is by fasting up testing throughout defect prediction.

In defect prediction process we guess which modules in the software are the most possible to be defected. It can speed up testing and improve its efficiency. Software defect prediction can be done by inspecting into software attributes and formalizing defect prediction models from these attributes. Such prediction models can be strongly formalized when enough attributes data is available in software repository to extract them (Azeem and Usmani, 2011; Saba and Altameem 2013).

The data mining techniques are used for software defects prediction (Kaur and Pallavi, 2013; Okutan and Yildiz, 2012; Lessmann, 2008; Azeem and Usmani, 2011, Saba et al., 2012). Data mining is process of observing hidden relationship between data and analyzing data from different perspective.

This paper is organized as follows: section 2 discusses the main classification techniques that will be used. The related works will be discussed in section 3. Section 4 clarifies the dataset and the experimental results. Conclusion and future works are given in section 5.

2. Approaches to classification

This section covers four existing approaches to classification: SVM, Decision Tree, KNN and NB algorithms. SVM is one of the effective algorithms that perform classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories (Joachims, 1999, Rehman and Saba, 2012). KNN is a statistical classification approach, which has been intensively studied in pattern recognition over four decades. KNN has been successfully applied in many fields, and showed promising results if compared with other statistical approaches such as Baysian based Network (Yang, 1999; Rahim et al., 2011; Saba et al., 2011). Decision Tree approach starts by selecting an attribute as a root node, and then it makes a branch for each possible level of that attribute. This will split the training instances into subsets, one for each possible value of the attribute. The same process will be repeated until all instances that fall in one branch have the same classification or the remaining instances cannot be split any further (Quinlan, 1993). NB which is a simple probabilistic classifier based on Baye's theorem.

The next four subsections describe the general nature, process for classifier training, advantages and disadvantages, of four learning methods that we considered.

2.1 K-**nearest Neighbor (KNN)**

KNN (Yang, 1999) is a statistical classification approach, which has been intensively studied in pattern recognition over four decades. KNN has been successfully applied to software defects, i.e. (Lessmann, 2008; Saba and Rehman, 2012), and showed promising results if compared with other statistical approaches such as Baysian based Network.

The KNN algorithm is quite simple: Given training and testing projects, the algorithm finds the k-nearest neighbors among the training projects, and uses the categories of the k-neighbors to weight the category of the test project. The similarity scores of each neighbor project to the test project are used as a weight of the categories of the neighbouring project. If several k-nearest-neighbors share a category, then the pre-neighbor weights of that category are added together, and the resulting weighted sum is used as the likelihood score of that category with respect to the test project. By sorting the scores of the candidates' categories, a ranked list is obtained for the test project. See (Yang and Liu, 1999; Yang, 1999) for further details.

2.2 **Naïve Bayesian (NB)**

The NB (Thabtah et al., 2009; Hadi et al., 2008b) is a simple probabilistic classifier based on applying Baye's theorem (Duda and Hart, 1973; Elarbi-Boudihir et al., 2011), and its predictive, easy and language independent method. When the NB classifier is applied on the software defects problem we use equation 1.

$$p(\text{class} \mid \text{project}) = \frac{p(\text{class}).p(\text{project} \mid \text{class})}{p(\text{project})} \qquad (1)$$

Where:

P (class|project): The probability that a given project D belongs to a given class C. P (project): The probability of a project, we can notice that p(project) is a constance divider to every calculation, so we can ignore it. P (class): The probability of a class, we can compute it from the number of projects that belong to a category divided by number of projects in all categories. P(project|class) represents the probability of project given class, and projects can be modelled as sets of words, thus the p(project|class) can be written like:

P(attribute|class**) = (nc + mp) /(n + m) (2)**

Where:

n = the number of training examples for which class = classj

nc = number of examples for which class = classj and attribute = attributei

p = attribute priori estimate for **P (**attribute|class**)**

m = the equivalent sample size.

NB has been successfully applied to software defects problem and showed good results if compared with other software defects techniques (Lessmann, 2008).

2.3 **Decision Trees**

The most popular Decision Tree learning program is C4.5 (Quinlan, 1993). This approach starts by selecting the best attribute as a root node, where each branch of the root corresponds to one of its possible value. The process is then repeated on each branch until no examples are left in the training data set. To decide which attribute to be selected at each step, IG (information gain) (Quinlan, 1986) is used. The attribute with the highest gain is chosen as the node. Informally, IG measures how well an attribute separates the training set with respect to class labels. Therefore, the higher the gain, the better separation resulting from classifying training examples on the associated attribute. For a formal concept, equations for computing information gain and an illustrative example see (Mitchell, 1997). (Lessmann, 2008) applied C4.5 and other classification methods in NASA data sets, and the results showed that the C4.5 produced competitive results if compared with other methods such as KNN, SVM, and Rocchio (Lessmann, 2008).

2.4 **Support Vector Machine (SVM)**

SVM was introduced by (Vapnik, 1995) as a class of supervised machine learning techniques. It is based on the principle of structural risk minimisation. In linear classification, SVM creates a hyper plane that separates the data into two sets with the maximum-margin. A hyper plane with the maximum-margin has the distances from the hyper plane to points when the two sides are equal. Mathematically, SVMs learn the sign function $f(x) = \text{sign}(wx + b)$, where w is a weighted vector in R^n. SVMs find the hyper plane $y = wx + b$ by separating the space R^n into two half-spaces with the maximum-margin. Linear SVMs can be generalised for non-linear problems. To do so, the data is mapped into another space H and we perform the linear SVM algorithm over this new space. Recently SVM has been successfully used on software defects (Lessmann, 2008) and they derived better results than other machine learning techniques such as NB, Decision Trees, and KNN with reference to accuracy.

3. literature review

In (Kaur and Pallavi, 2013), they discussed how three different data mining techniques (clustering, classification, and association) can be used in software defect production, they delimited those techniques arguing that the three different techniques have given different results on different data sets and concluding that a certain techniques can perform differently under different data sets.

In (Okutan and Yildiz, 2012), they surveyed some defect prediction technique raising main research question in the field such as, how to assess a prediction technique or, which repositories and datasets are best to mind. The authors also argued that software prediction model only works well when enough data is available in software repository within the organization to initially feed the model. And their most important finding is that there is no single data mining technique that is best or suitable for all type of software projects, In order to select a better data mining algorithm, domain expert must consider the various factors like problem domain, type of data sets, nature of project (Rehman and Saba, 2011).

In (Lessmann, 2008), they used Bayesian networks to study the relation between software metrics and defect proneness. It is important to select a related and representative set of metrics in such a study as many software metrics are available. The authors selected a set of good important metrics and tried to find which of them are probably most related to the existence of errors in software. After mining into nine different data sets they concluded with good certainty that some of the related metrics are the response for class (RFC) and line of code (LOC), and some of the unrelated metrics are the number of children (NOC) and the depth of inheritance tree (DIT).

4. data set and experimental results

4.1 Dataset

The data used in our experiments are The software defects data sets (NASA, 2004), the data set consist of 498 projects that belongs to 2 categories, the categories are false (no defects) and true (defects), Table 1 represent the number of projects for each category.

Table 1: Number of projects per Category

Category Name	Number of projects
False	449
True	49
Total	**498**

For each project there are 22 attributes divided to 5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, a branch-count, and 1 goal field) as shown in table 2.

Table 2: Description of all attributes (NASA,2004)

Name of attribute	description
loc	numeric % McCabe's line count of code
v(g)	numeric % McCabe "cyclomatic complexity"
ev(g)	numeric % McCabe "essential complexity"
iv(g)	numeric % McCabe "design complexity"
n	numeric % Halstead total operators + operands
v	numeric % Halstead "volume"
l	numeric % Halstead "program length"
d	numeric % Halstead "difficulty"
i	numeric % Halstead "intelligence"
e	numeric % Halstead "effort"
b	numeric % Halstead
t	numeric % Halstead's time estimator
lOCode	numeric % Halstead's line count
lOComment	numeric % Halstead's count of lines of comments
lOBlank	numeric % Halstead's count of blank lines
lOCodeAndComment	numeric
uniq_Op	numeric % unique operators
uniq_Opnd	numeric % unique operands
total_Op	numeric % total operators
total_Opnd	numeric % total operands
branchCount	numeric % of the flow graph
defects	{false,true} % module has/has not one or more. reported defects

4.2 **Experimental Results**

We used three evaluation measures (Recall, Precision, and F1) as the bases of our comparison, where F1 is computed based on the following equation:

$$F1 = \frac{2 * Precision * Recall}{Recall + Precision} \quad (1)$$

Precision and recall are widely used evaluation measures in IR and ML, where according to Table 2

$$Precision = \frac{a}{(a+b)} \quad (2)$$

$$Recall = \frac{a}{(a+c)} \quad (3)$$

Table 2 Cases possible sets based on a classification in ML

Iteration	Predicated as actual Class	Predicted as other classes
Actual Class	A	c
Other Classes	B	d

Table 3 gives the F1, Recall, and Precision results generated by the four classifiers (NB, SVM, Decision Tree and KNN) against software defects data sets where in each data set using ten-fold cross-validation.

Table 3: F1, Recall, and Precision results generated by four classifiers (NB, SVM, Decision Tree, and KNN)

Name of algorithm	**measure**	**True**	**False**	**Average**
Naïve Bayes	Precision	0.286	0.925	0.862
	Recall	0.327	0.911	0.853
	F1	0.305	0.918	0.858
SVM	Precision	0	0.901	0.812
	Recall	0	0.993	0.896
	F1	0	0.945	0.852
Decision Tree	Precision	0.176	0.904	0.833
	Recall	0.061	0.969	0.88
	F1	0.091	0.935	0.852
KNN	Precision	0.186	0.91	0.839
	Recall	0.163	0.922	0.847
	F1	0.174	0.916	0.843

Cross validation is a known evaluation method in data mining, where the training data is divided randomly into *n* blocks, each block is held out once, and the classifier is trained on the remaining *n*-1 blocks; then its error rate is evaluated on the holdout block. Therefore, the learning procedure is executed *n* times on slightly different training data sets. All the experiments were conducted using the Weka open source software (Weka software, 2001).

After analyzing Table 3 we found that the SVM Classifier outperformed NB, Decision Tree and KNN on false data set with regards to F1 results as shown in figure 1.

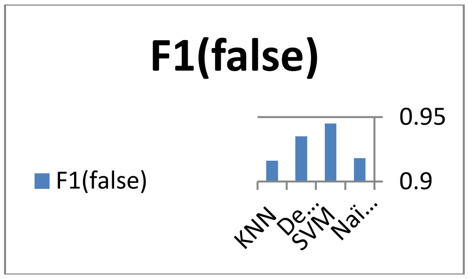

Figure 1: F1 measure in false data set for KNN, Decision Tree, SVM, and Naïve Bayes.

Precision results obtain that the NB outperformed SVM, Decision Tree and KNN on False data set as shown in figure 2. Also Recall results obtain that the SVM outperformed NB, Decision Tree and KNN on false data set as shown in figure 3.

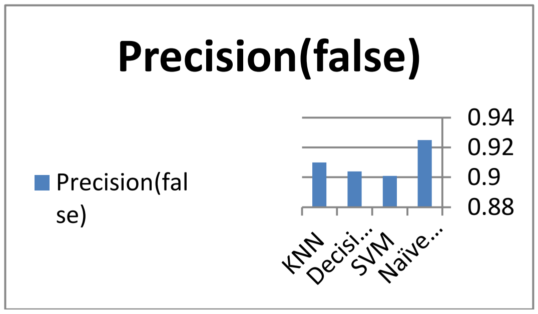

Figure 2: Precision measure in false data set for KNN, Decision Tree, SVM, and Naïve Bayes.

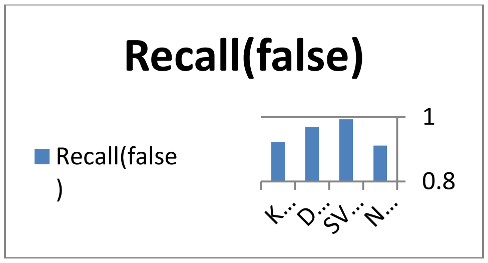

Figure 3: Recall measure in true data set for KNN, Decision Tree, SVM, and Naïve Bayes.

The average of three measures obtained against false data set indicated that the NB classifier outperformed all algorithms. But all classifiers performed poor in true data set, because the distribution of the data sets not balanced i.e. 90% false data set and 10% for true data set as shown in figures 4, 5, and 6.

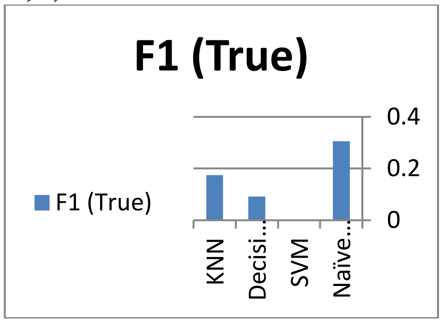

Figure 4: F1 measure in true data set for KNN, Decision Tree, SVM, and Naïve Bayes.

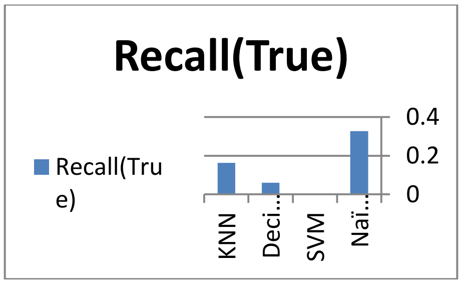

Figure 5: Recall measure in true data set for KNN, Decision Tree, SVM, and Naïve Bayes.

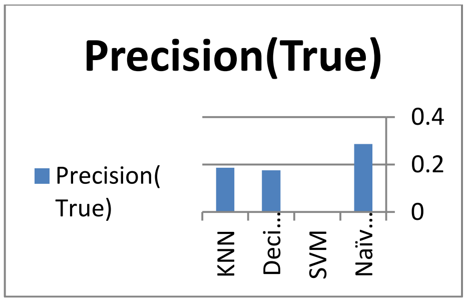

Figure 6: Precision measure in true data set for KNN, Decision Tree, SVM, and Naïve Bayes

Finally, all classifiers perform excellent to detecting defects on the software. This excellent result may indicate that data mining can be considered as a tool to determine if the software has defects or not as shown in figures 7, 8, and 9.

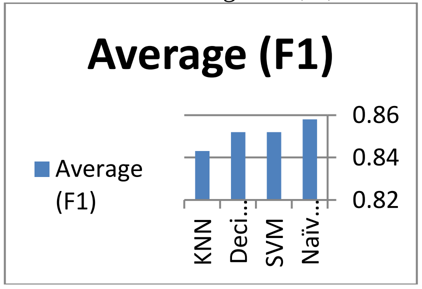

Figure 7: F1 average for KNN, Decision Tree, SVM, and Naïve Bayes.

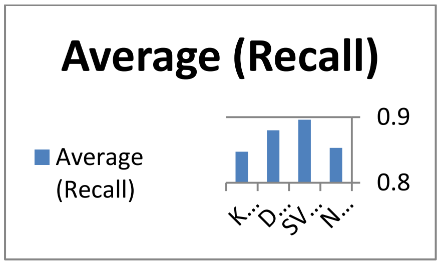

Figure 8: Recall average for KNN, Decision Tree, SVM, and Naïve Bayes.

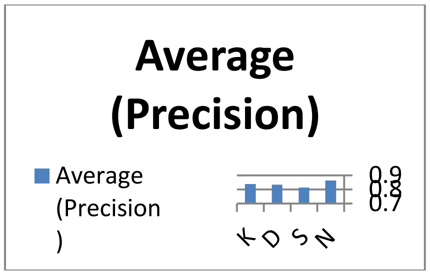

Figure 9: Precision average for KNN, Decision Tree, SVM, and Naïve Bayes

Conclusion and future works

In this paper we discussed the problem of software defects prediction and we used classification techniques to solve this problem. We used the NB algorithm which is based on probabilistic framework, SVM algorithm, K-nearest Neighbor (KNN) and Decision Trees to handle our problem.

Performance of All classifiers was excellent to detecting defects on the software. This performance may indicate that data mining can be considered as a tool to determine if the software has defects or not. The average of three measures obtained against NASA data set indicated that the NB algorithm outperformed the others.

In near future, we intend to propose a new multi-label classification approach based on association rule for the software defects problem and enhance one of the classification algorithms.

References

1. Azeem, N. and Usmani S.(2011) analysis of data mining based software defect prediction techniques, global journal of computer science and technology, volume 11(16), 2011.
2. A. Rehman and T. Saba (2012). "Evaluation of Artificial Intelligent Techniques to Secure Information in Enterprises". Artificial Intelligence Review, DOI. 10.1007/s10462-012-9372-9.
3. Duda R. and Hart P. (1973). Pattern classification and scene analysis. John Wiley & son.

4. Elarbi-Boudihir, M. A. Rehman and T.Saba (2011). “Video Motion Perception Using Operation Gabor Filter”. International Journal of Physical Sciences, Vol. 6(12), pp. 2799-2806.
5. Hadi W., Thabtah F., ALHawari S., Ababneh J. (2008) Naive Bayesian and K-Nearest Neighbour to Categorize Arabic Text Data. Proceedings of the European Simulation and Modelling Conference. Le Havre, France,(pp. 196-200), 2008.
6. Saba T, Al-Zaharani S, Rehman A. Expert System for Offline Clinical Guidelines and Treatment Life Sci Journal 2012;vol. 9(4):2639-2658
7. Saba, T. Altameem, A. (2013) Analysis of vision based systems to detect real time goal events in soccer videos, Applied Artificial Intelligence 27(7), 656-667
8. Joachims T. (1999). Transductive Inference for Text Classification using Support Vector Machines. Proceedings of the International Conference on Machine Learning (ICML), (pp. 200-209).
9. Kaur, P. and Pallavi. (2013) Data mining techniques for software defect prediction, international journal of software and web sciences (IJSWS). 12-347.
10. Rehman, A. and Saba, T. (2011). “Document Skew Estimation and Correction: Analysis of Techniques, Common problems and Possible Solutions” Applied Artificial Intelligence, Vol. 25(9), pp. 769-787.
11. Lessmann, S. (2008) Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. IEEE transactions on software engineering vol 34, NO 4.
12. Mitchell M. (1997). Machine Learning, chapter IV, Artificial Neural Networks, (pp. 81-127). WCB/McGraw-Hill, New York, New York.
13. NASA, CM1/software defect prediction, http://promise.site.uottawa.ca/SERepository/datasets/cm1.arff, 2 December 2004.
14. T. Saba, A. Rehman (2012). Machine Learning and Script Recognition, Lambert Academic publisher, pp:99-117.
15. Okutan, A. and Yildiz O. (2012) software defect prediction using Bayesian networks, Springer science, 01 august 2012.
16. Quinlan J. (1986). Induction of decision trees. Machine Learning, 1(1986): 81 – 106.
17. Quinlan, J. (1993) "C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.
18. Thabtah F., Eljinini M., Zamzeer M., Hadi W. (2009) Naïve Bayesian based on Chi Square to Categorize Arabic Data. In proceedings of The 11th International Business Information Management Association Conference (IBIMA) Conference on Innovation and Knowledge Management in Twin Track Economies, Cairo, Egypt 4 - 6 January. (pp. 930-935).
19. Vapnik V. (1995). The Nature of Statistical Learning Theory, chapter 5. Springer-Verlag, New York.
20. WEKA. Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka, 2001.
21. Yang Y. and Liu X. (1999). A re-examination of text categorization methods, Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), (pp. 42-49).
22. Yang, Y., and Pedersen, J.O. (1997) "A comparative study on feature selection in text categorization," In Proc. of Int'l Conference on Machine Learning (ICML), pp. 412-420.
23. MSM Rahim, A Rehman, MFA Jabal, T Saba (2011) Close Spanning Tree (CST) Approach for Error Detection and Correction for 2D CAD Drawing, International Journal of Academic Research, Vol. 3(4), pp. 525-533
24. T. Saba, A. Rehman, M. Elarbi-Boudihir (2011). Methods and strategies on off-line cursive touched characters segmentation: a directional review, Artificial Intelligence Review, DOI 10.1007/s10462-011-9271-5.pp:45-54.

11/23/2013