Software Defect Classification using Bayesian Classification Techniques

Sakthi Kumaresh
Dept. of Computer Science
MOP Vaishnav College for Women, Chennai, India

Baskaran R
Dept. of Computer Science & Engineering Anna University
Guindy, Chennai, India

Meenakshy Sivaguru
Dept. of Computer Science
MOP Vaishnav College for Women, Chennai, India

ABSTRACT
Classifying a defect is an important activity for improving software quality. It is important to classify defects as they contain information regarding the quality of processes and product. The information gathered from defects can be used to track the future projects, and to improve its processes. Considering the need to classify the defect and to gain insight knowledge of defect details, this paper attempts to analyze software defect using Bayes net and Naïve Bayes Classification techniques.

It is very difficult to produce defect free software product, however, the main purpose of any software engineering activity is to prevent defects from being introduced in the first place. As fixing of software defects are expensive and time consuming, various defect prediction techniques and defect tracing mechanism are being used to prevent software defects from occurring.

The aim of this paper is to show the comparative analysis of software defect classification using Bayes net and Naïve Bayes classification techniques in terms of accuracy, Precision, Recall etc. The Naive Bayesian classifier assumes that all variables contribute toward classification and that they are mutually independent. A Naive Bayesian model leads to a simple prediction framework that gives good result which makes it particularly useful for very large datasets like public NASA MDP repository and the same is experimented in this study. The study revealed that, the performance of Naïve Bayes classification for software defect outperforms Bayes net classification method.

Keywords
Defect, Bayesian classification, Defect Prediction, Software quality, Naïve Bayes.

1. INTRODUCTION
A software defect can be defined as imperfections in software development process that would cause software to fail to meet the desired expectations [1]. Detecting and correcting these defects is one of the important tasks in software development activities. It will not be possible to eliminate all defects but it is possible to minimize the number of defects and their severe impact on the projects. To do this a defect preventive process needs to be implemented that focuses on improving software quality by decreasing the defect counts. A little investment in predicting software defects and providing corrective measures can yield significant returns.

In Software development process, testing of software is the main phase which identifies the defects of the software. If a developer or a tester can identify the software defects properly then, it reduces the cost, time and effort involved in detecting and correcting the defects. Developing high-quality software systems is a challenging and very expensive task. As the software development process is a human intensive activity, it is impossible to produce defect free software [4]. To deliver defect free software, it is necessary to identify and fix the defect as many as possible before the product delivers to the customers.

Software defect prediction has been an important topic of research in the field of software engineering for more than a decade. Studies on software defect prediction aim to estimate the probability a software module containing errors. Public data repositories like NASA data repository are used for building the classification model. Sample module characteristics of this data set include Line of Code (LOC), Halsted measures and McCabe Measures.

Bayes classification, rule based classification like association rule mining, tree-based methods such as ID3, C4.5, RIPPER, Neural Networks (NN), Support Vector Machines (SVM) etc. are some of the classification methods available to build software prediction models.

This research work aims to evaluate the performance of supervised machine learning techniques on analyzing defect in the software dataset through two of the classification models namely Bayes net and Naïve Bayes classification. We studied the performance of two classification algorithms on seven publicly available datasets from the NASA MDP Repository. This paper emphasizes on the performance of classification algorithms in categorizing seven datasets (CM1, JM1, KC1, KC2, MC1, MW1 and PC1) under two classes namely Defective and Normal. Naïve Bayes Classification algorithm produced greater accuracy when compared to Bayes net in classifying the datasets and hence the features selected by this technique were considered to be the most significant features.

2. MINING SOFTWARE REPOSITORY
Data Mining is the process of finding meaningful information and patterns from large sets of databases. Data Mining uses various techniques such as Frequent Pattern Mining, Pattern Matching, Clustering and Classification [5]. These algorithms and techniques are applied on software defect repository to analyze software defect data. Software defect repository contains data obtained from a defect tracking system on a major project. These data can be used to analyze whether the software module is defect prone or not. Data mining techniques has significant influence on information discovery from this software defect data, as it helps the software developers to improve the quality of software. It is aimed to determine whether software module has a higher failure risk or not.

3. PROPOSED WORK

Most of the current software defect prediction techniques have something to do with software metrics and classification algorithm. In this work, public NASA MDP data sets are used to build prediction models based on Bayes net and Naïve bayes and then comparison is made on the performance of these two models. Later, these models can also be used in future version of software to predict software defects on different granularity.

3.1 Bayesian Classification

Bayesian Classifiers are statistical classifiers. The probability that a given record in the dataset belongs to a particular class can be predicted with the help of Bayesian classifiers, using the class membership probabilities. Naïve Bayes classification is based on Baye's theorem. Here the optimal rules were given as input to the naïve bayes classifier. This type of classifier has the advantage that it is easy to implement and generate good results. Studies comparing classification algorithms have found naïve Bayesian classifier to be comparable in performance with decision tree and selected neural network classifiers.

Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$P(c|x) = P(x|c)\ P(c) /\ P(x) \quad [13]$$

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \ldots\ldots\ldots\ldots\ldots\ldots x\ P(x_n|c) \times P(c)$$

- P(c|x) is the posterior probability of class (target) given predictor (attribute).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor [6][7][8][12].

The posterior probability can be calculated by first, constructing a frequency table for each attribute against the target. Then, transforming the frequency tables to likelihood tables and finally uses the Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction [7].

3.2 Bayes Net

Bayes net is a graphical representation of a set of random variables and their conditional dependencies via a directed acyclic graph. Here, the random variables are denoted by nodes and their conditional dependencies are denoted by connecting edges. The edges can be directed or undirected. If there is no edge between two nodes, that means they are conditionally independent whereas if there is an edge, that means they are conditionally dependent. Bayes net is often termed as belief networks or causal networks [10].

A natural way to measure how well Bayesian network performs on a given data set is to predict its future performance by estimating classification accuracy. Cross validation provides an out of sample evaluation method to facilitate this by repeatedly splitting the data in training and validation sets. A bayesian network structure can be evaluated by estimating the network's parameter from the training set and the resulting bayesain network performance determined against the validation set. The average performance of the bayesian network over the validation sets provides a metric for the quality of the network [11].

4. EXPERIMENTAL EVALUATION

This defect prediction study was tested on 7 out of 12 data sets from the public NASA MDP repository. Each data set is comprised of a number of software modules (cases), the corresponding number of defects and various static code attributes. The software module in the data set is characterized by Lines of Code (LOC) based metrics, Halstead metrics and McCabe complexity measures. The data sets in NASA MDP repository are part of various space exploration related software projects such as flight software for an earth orbiting satellite (PC2), storage management system for ground data (KC1 and KC2) [3], Database software (MW1), NASA spacecraft system (CM1). The software was written in software languages like C, C++ and JAVA. Experimentation of data sets for software defect classification is done with WEKA Data-mining Tool. For testing the accuracy of the classification we use leave one out cross validation (LOOCV) technique with k=10[2][9]. Sample output for the defect data set CM1 is shown in Figure 1.

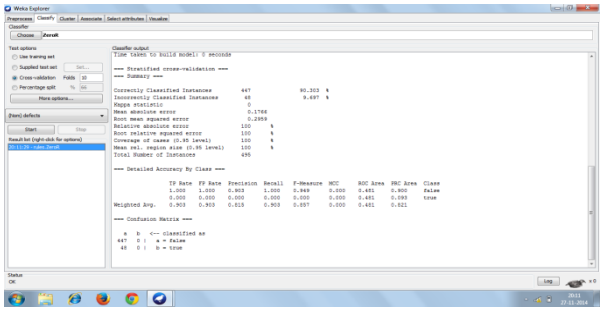

Fig 1: Sample Output from WEKA for the data set CM1

Table 1 summarizes the dataset and their properties. After the prediction process, a confusion matrix was designed. This matrix would be useful in the process of performance evaluation.

Table 1: Data Set Characteristics

Data Set	Attributes	Module	Defects
CM1	21	505	48
JM1	22	10878	2102
KC1	22	2105	325
KC2	22	498	105
MC1	39	4621	68
MW1	38	403	31
PC2	37	4505	23

Table 2 illustrates a confusion matrix for binary class problem having positive and negative class values. Based on the confusion matrix (TP, TN, FN, FP), we calculate the various performance measures namely Accuracy, Sensitivity and Specificity etc.

Table 2: Confusion Matrix

		Predicted Class	
Actual Class		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- TP – Correctly classified instance as defect prone.
- TN – An instance correctly classified as not defect prone.
- FN – Defect prone instance is incorrectly classified as not defect prone instance
- FP – An instance is classified as defect prone, but actually it is not

5. RESULTS AND DISCUSSION

Table 3 summarizes the performance measures for various data sets under two classification techniques Bayes Net and Naïve Bayes.

Table 3: Performance Measures

Measures	Methods	CM1	JM1	KC1	KC2	MC1	MW1	PC2
Accuracy %	Bayes Net	63.63	70.71	70.64	78.92	86.11	86.56	89.84
	Naïve Bayes	84.84	81.31	82.24	83.67	93.89	83.58	97.18
Sensitivity	Bayes Net	0.75	0.553	0.726	0.73	0.794	0.483	0.608
	Naïve Bayes	0.354	0.205	0.372	0.48	0.602	0.516	0.391
Specificity	Bayes Net	0.624	0.741	0.702	0.804	0.861	0.897	0.899
	Naïve Bayes	0.901	0.949	0.905	0.929	0.941	0.899	0.974

5.1 Accuracy

Accuracy contributes to, precisely, measure the percentage of predictions that are correct [8]. It is also called the correct classification rate. Accuracy is the basic index in defect prediction. Our study reveals that accuracy measure is high for Naïve Bayes classification than Bayes net and it is shown graphically in fig 2.

Accuracy = (TP + TN) / (TP+FP+FN+TN)

From Accuracy, we can calculate the error rate.

Error Rate = (1-Accuracy)

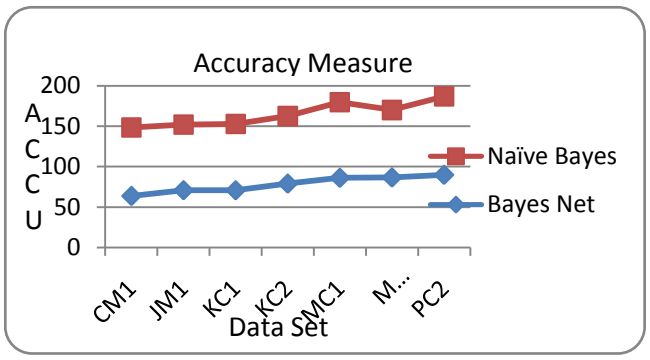

Fig 2: Comparison of Accuracy Values

5.2 Sensitivity

Sensitivity is the measure used to report how effective our classification algorithm is in identifying instances with a defect. In our study sensitivity is higher in the case of Bayes net than Naïve Bayes in all the data sets that we have experimented.

Sensitivity = Recall = TP / (TP + FN)

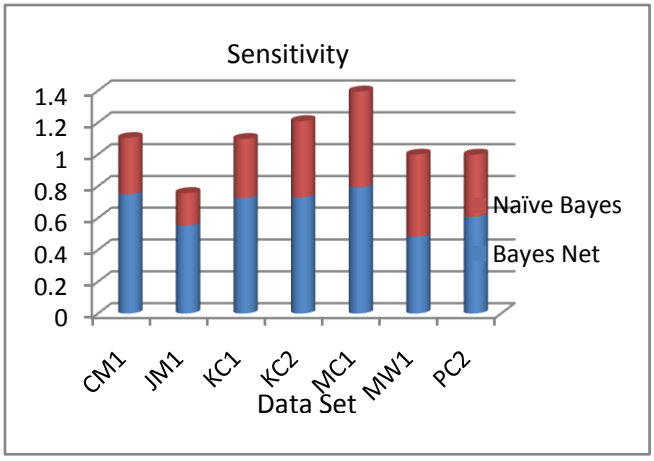

Fig 2: Comparison of Sensitivity measure

5.3 Specificity

The Specificity highlights the measure that is used to report how effective our classification algorithm is, in pointing out instances without a defect [8]. When the specificity is very low, many non-defect-prone modules could be classified as the defect prone modules and be tested and verified, which increases the cost in time and investment. Our study reveals that the specificity values of Bayes net are lower than Naïve bayes. In general, Sensitivity and specificity values are inversely proportional. If the sensitivity is high, the specificity value will be low and vice versa and the same is evident in our study. In Bayes net classification, sensitivity is high and specificity is low whereas in Naïve Bayes Sensitivity is low and Specificity is high.

Specificity = TN / TN + FP [14]

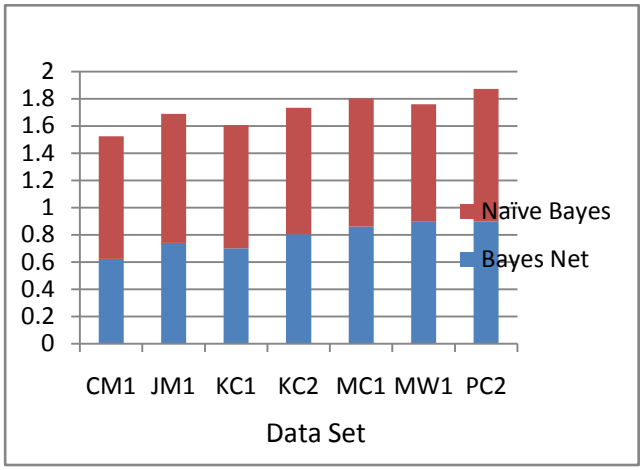

Fig 3: Comparison of Specificity measure

5.4 Precision

The precision measures the ratio of the correctly classified positive modules to the set of the positive modules. It is also called true positive Rate of consistency. When the percentage of correctly predicted positive modules is low or the percentage of incorrectly classified as negative modules is high, a low accuracy is caused. Thus the precision is also an important index.

$$\text{Precision} = (TP) / (TP + FP)$$

5.5 Recall

The recall is the percentage of the correctly predicted positive modules in the whole modules with defects. It is generally presumed, that the higher is the recall, the less is the error-prone modules not found. When the recall is high, it means that, the classifier has a higher probability to find the positive modules, but may be on the cost of classifying the negative modules as the positive modules which causes a low specificity.

$$\text{Recall} = TP / (TP + FN)$$

Table 4: Parameter Values for Classification Models

Modules	Method	Precision	Recall
CM1	Bayes Net	0.176	0.75
	Naïve Bayes	0.279	0.354
JM1	Bayes Net	0.325	0.553
	Naïve Bayes	0.478	0.205
KC1	Bayes Net	0.31	0.726
	Naïve Bayes	0.419	0.372
KC2	Bayes Net	0.493	0.73
	Naïve Bayes	0.64	0.48
MC1	Bayes Net	0.998	0.862
	Naïve Bayes	0.075	0.603
MW1	Bayes Net	0.954	0.898
	Naïve Bayes	0.283	0.484
PC2	Bayes Net	0.026	0.609
	Naïve Bayes	0.062	0.391

6. CONCLUSION AND FUTURE WORK

Defects that originate during software development process, lead to poor quality software which might be the cause of software failure. Software quality largely depends on prediction of software defect. In this work, in order to predict the defects in a software development process, Bayesian classification techniques are used.

Implementation of data mining techniques to analyze and predict software defect from large number of data set is done to improve the efficiency and quality of software development. This research work has investigated the performance of Bayes net and Naive Bayes defect classification method for software defect prediction problem. It also analyzes the experimental results using WEKA and measures the performance of both classifiers through several performance metrics such as accuracy, sensitivity and specificity. Experiment result shows that Naïve Bayes classification technique applied to software defect prediction could achieve better accuracy than Bayes net classifiers.

Future work would involve multinomial text classification using naïve bayes classification technique. As the Software defects are introduced during various stages of software development, data records in defect data set may fall into more than two classes. Moreover in certain data set, defect data is available in text form, hence, the multinomial text classification model. When such a model is developed, defects that fall into various categories can be predicted, necessary preventive action can be taken to avoid the recurring of such defects in future projects. Therefore, the cost and time involved in detecting and correcting the bugs can be minimized, improving the software development quality.

7. REFERENCES

[1] Sakthi Kumaresh, R.Baskaran, "Defect Analysis and Prevention for Software Process Quality Improvement" Published in International Journal of Computer Applications. vol. 8 – No 7, October 2010.

[2] D Rodriguez, R Ruiz, M Garre, "Attribute Selection in Software Engineering Datasets for detecting Fault Modules" in 33rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2007) 0-7695-2977—1/07 IEEE.

[3] Baojun, Ma, Karel Dejaeger, Jan Vanthienen, Bart Baesens, "Software Defect Prediction based on Association rule classification" K U Leuven , http://ssrn.com/abstract=1785381

[4] Naheed Azeem, Shazia Usmani, "Analysis of Data Mining Based Software Defect Prediction Techniques" in Global Journal of Computer Science and Technology" Volume 11, Issue 16, September 2011.

[5] Maninderjit Kaur, Dr. Sushil Kumar Garg, 'An approach to Detect and Classify Bugs Using Data Mining Techniques" published in International Journal of

Advanced Research in Computer Science and Software Engineering" , volume 4,Issue 7, July 2014.

[6] M. Surendra Naidu, Dr. N. Geethanjali "Optimal Rule Selection Based Defect Classification System using Naïve Bayes Classification" published in International Journal of Engineering Science and Technology, Volume 5, No 08, August 2013.

[7] Hui Wang, "Software Defects Classification Prediction Based on Mining Software Repository" published in Uppsala University, January 2014

[8] Dulal Chandra Sahana, "Software Defect Prediction Based on Classification Rule Mining" Dissertation submitted at National Institutue of Technology, Rourkela, May 2013.

[9] Tao Wang, Weihua Li et al, "Software Defect Prediction Based on Classifiers Ensemble" published in "Journal of Information & Computational Sciences" 8:16 (2011) http://www.joics.com

[10] Mohammad Masudur Rahman, Shamima Yeasmin, 'Adaptive Bug Classification for CVE list using Bayesian Probabilistic Approach' Technical Report submitted in University of Saskatchewan, Canada.

[11] Remco R Bouckaert, Bayesian Network Classifiers in Weka for version 3-5-7, May 2008

http://www.cs.waikato.ac.nz/~remco/weka.bn.pdf

[12] Ruchika Rana, Jyothi Pruthi "Naïve bayes classification" published in International Journal for specific research and development. http://www.ijsrd.com/articles/IJSRDV2I4378.pdf

[13] "Naïve Bayesian" http://www.saedsayad.com/naive_bayesian.html

[14] Cagatay Catal, Banu Diri "A Fault detection Strategy for Software Projects" ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) UDK/UDC 004.416.052.42 http://hrcak.srce.hr/file/143476