Empirically Validating Software Metrics for Risk Prediction Based on Intelligent Methods*

Zhihong Xu[1], Xin Zheng[1], Ping Guo[2], Senior Member, IEEE
[1] Image Processing and Pattern Recognition Laboratory
Beijing Normal University, Beijing 100875
China.
[2] School of Computer Science
Beijing Institute of Technology, Beijing 100081, China
Image Processing and Pattern Recognition Laboratory
Beijing Normal University, Beijing 100875
China.
pguo@ieee.org

Journal of Digital Information Management

ABSTRACT*: The software systems which are related to national science and technology projects are very crucial. This kind of systems always involves high technical factors and has to spend a large amount of money, so the quality and reliability of the software deserve to be further studied. Hence, we propose to apply four intelligent classification techniques most used in data mining fields, including Bayesian belief networks (BBN), nearest neighbor (NN), rough set (RS) and decision tree (DT), to validate the usefulness of software metrics for risk prediction. Results show that comparing with metrics such as Lines of code (LOC) and Cyclomatic complexity (V(G)) which are traditionally used for risk prediction, Halstead program difficulty (D), Number of executable statements (EXEC) and Halstead program volume (V) are the more effective metrics as risk predictors. By analyzing obtained results we also found that BBN was more effective than the other three methods in risk prediction.*

Categories and Subject Descriptors
D. 2.4[Software/Program Verification**]; D 2.8 [Software Metrics]** Multimedia Databases; **D.2.9 [Software Management]**

General Terms
Software reliability, Software systems, Software quality and reliability

Keywords: Software metrics, Bayesian belief networks, decision tree, nearest neighbor, risk prediction**.**

Received 30 November 2006; Revised 15 February 2007; Accepted 12 March 2007

1. Introduction

The software systems of national science and technology projects are very important all the time. Quite different from commercial software systems developed in company, they are full of new developed high technology under investigation and always used to implement some specific tasks which are very significant and have strict requirement for the reliability and quality of these systems. So the reliability and quality of the software needs to be seriously considered.

Various kinds of code measurements can be quite useful in obtaining information about the risk of the software. In the early period of software testing, there are no other information than software metrics which are a crucial source of information for decision-making and can help us have an understanding of the risk of the software system, so we can use the metrics to analyze the system, predict its risk, and then calculate the time needed for software testing or determine whether the whole system can be released or not.

* This work was supported by the grants from the National Natural Science Foundation of China (Project No. 60275002, 60675011).
[2] To whom all communications should be addressed

Many researchers have analyzed the relationships between software metrics and the quality of the software systems. Ping Guo et al. [9] used Expectation-Maximum (EM) algorithm to develop a model for predicting software quality by only employing software size and complexity metrics. Gyimothy et al. [10] as well as Basili et al. [11] both have done researches on the object-oriented metrics and got the similar results. Besides logistic regression used by Basili et al. [11], Gyimothy et al. [10] applied machine learning techniques (decision trees and neural networks). Other researchers used different methods. All in all, the methods they used fall into the following main categories: association analysis [13], clustering analysis [9], [14], classification and regression analysis [10, 11, 15].

In order to find some relationships between the risk of the software and the metrics extracted from the code of the national project related software system, in this paper, we applied four classical intelligent classification methods which were most used in data mining fields, namely Bayesian belief networks (BBN) [1, 2, 3], nearest neighbor (NN) [4] rough set (RS) and decision tree (DT) [5], [6], to predict the risk of the core source code of a spectrum processing software system which is a “863” National Project related to the Large Sky Area Multi-Object Spectroscopic Telescope (LAMOST) Project in China. In this paper, we mainly studied core algorithms of this software system which was coded with C++ language, and studied the risk of every function in the software system. In order to perform our analysis, we have taken complete records during the software development process, especially in the test stage. A metric extraction tool named Krakatau Professional [7], which was a fully–featured software metric tool, designed for source code quality and software metrics specialists, was adopted in the analysis. Krakatau Professional for C/C++ are industry strength software metrics tool for organizations committed to delivering maximum quality and can extract the metrics of projects, files and functions.

It was found when comparing with extracted metrics such as Lines of code (LOC) and Cyclomatic complexity (V(G)) which were traditionally used for risk prediction [8], [9] in this kind of software systems, Halstead program difficulty (D), Number of executable statements (EXEC) and Halstead program volume (V) were more useful. At the same time, by data analyzing we also found that BBN was more effective than the other three methods in risk prediction.

The organization of this paper is as follows: In the next section, we will describe the metrics suite obtained from Krakatau Professional and also state our hypotheses about

their expected connection with risk. In Section 3, we will present the results of our experiments in applying Bayesian belief networks, nearest neighbor, rough set and decision tree models to risk prediction. In last section, we will present our conclusions and the considerations for future work.

2. Metrics

In this section, we will give the descriptions of the metrics obtained by Krakatau Professional tool. This tool can extract more than twenty metrics of functions, but we only chose twelve of them, which are easy to understand and commonly used for risk prediction. The detail descriptions of these twelve metrics are as follows:

- Code: CDENS

Name: Control density
Description: CDENS = CONTROL / NSTAT. Control density represents the percentage of control statements in the code.

- Code: CONTROL

Name: Number of control statements
Description: A count of the number of control statements in the function. Every occurrence of a control keyword (such as if, else, while, switch, case, etc.) causes this count to be incremented by 1.

- Code: D

Name: Halstead program difficulty
Description: Halstead measure of how compactly the function implements its algorithms. It shows how difficult this function's code is to understand.

- Code: V

Name: Halstead program volume
Description: Halstead volume for the function. Calculated as: V = Nlog2n.

- Code: EXEC

Name: Number of executable statements
Description: A count of the number of executable statements in the function. Every statement that is not a control statement (see CONTROL) causes this count to be incremented by 1.

- Code: LOC

Name: Lines of code
Description: Number of lines in this function, including source, white space and comments.

- Code: N

Name: Halstead program length
Description: Halstead program length (N) for this function is calculated as N1 + N2.
N1 = Total number of operators.
N2 = Total number of operands.

- Code: NSC

Name: Number of semicolons
Description: A count of the number of semicolons in this function excluding those within comments and string literals. This is useful for approximating "logical lines of code".

- Code: NSTAT

Name: Number of statements
Description: NSTAT = CONTROL + EXEC.

- Code: SLOC

Name: Source lines of code
Description: Number of source lines in this function, excluding white space and comments. This is not a count of semicolons or distinct statements, but a count of physical lines that contain source code.

- Code: V(G)

Name: Cyclomatic complexity
Description: Cyclomatic complexity (V(G)) is a measure of the complexity of the function's decision structure. It is the number of linearly, independent paths and therefore, the minimum number of paths that should be tested.

- Code: n

Name: Halstead program vocabulary
Description: Halstead program vocabulary (n) for this function is calculated as n1 + n2.
n1 = Number unique or distinct operators.
n2 = Number unique or distinct operands.

All the definitions of metrics are from the metrics extraction software. In order to validate the above metrics as quality indicators, their expected relationship with risk must be validated. The experimental hypotheses to be tested are, for each metric, as follows:

CDENS hypothesis: A function with higher density of control statement than its peers is more fault-proneness than they are. (Null hypothesis: A function with higher density of control statement than its peers is no more fault-proneness than they are.)

CONTROL hypothesis: A function with more number of control statements than its peers is more fault-proneness than they are. (Null hypothesis: A function with more number of control statements than its peers is no more fault-proneness than they are.)

D hypothesis: A function that implements its algorithms more compactly than its peers is more fault-proneness than they are. (Null hypothesis: A function that implements its algorithms more compactly than its peers are no more fault-proneness than they are.)

V hypothesis: A function with bigger volume than its peers is more fault-proneness than they are. (Null hypothesis: A function with bigger volume than its peers is no more fault-proneness than they are.)

EXEC hypothesis: A function with more number of executable statements than its peers is more fault-proneness than they are. (Null hypothesis: A function with more number of executable statements than its peers is no more fault-proneness than they are.)

LOC hypothesis: A function with more lines of codes than its peers is more fault-proneness than they are. (Null hypothesis: A function with more lines of codes than its peers is more fault-proneness than they are.)

N hypothesis: A function with higher number of all operands and operators than its peers is more fault-proneness than they are. (Null hypothesis: A function with higher number of all operands and operators is no more fault-proneness than they are.)

	Max	Min	Medium	Mean	Standard Deviation
CDENS	1	0	0.318	0.362	0.125
CONTROL	248	0	7	20.143	20.420
D	68.489	0	12.435	17.622	11.871
V	19607.9	0	504.227	1778.78	1868.648
EXEC	667	0	14	48.971	53.584
LOC	1220	4	45.5	107.1	105.583
N	2558	4	108.5	274.214	264.122
NSC	580	1	25.5	51.757	50.696
NSTAT	915	1	22.5	69.114	73.751
SLOC	1145	4	33	91.434	93.810
V(G)	197	1	6	16.371	16.498
n	213	3	35	49.543	30.150

Table 1 Discriptive Statistics of the Functions

NSC hypothesis: A function with more number of semicolons than its peers is more fault-proneness than they are. (Null hypothesis: A function with more number of semicolons than its peers is no more fault-proneness than they are.)

NSTAT hypothesis: A function with more number of statements than its peers is more fault-proneness than they are. (Null hypothesis: A function with more number of statements than its peers is no more fault-proneness than they are.)

SLOC hypothesis: A function with more number of source lines of codes than its peers is more fault-proneness than they are. (Null hypothesis: A function with more number of source lines of codes than its peers is no more fault-proneness than they are.)

V(G) hypothesis: A function with higher cyclomatic complexity than its peers is more fault-proneness than they are. (Null hypothesis: A function with higher cyclomatic complicity than its peers is no more fault-proneness than they are.)

n hypothesis: A function with higher number of unique operands and operators than its peers is more fault-proneness than they are. (Null hypothesis: A function with higher number of unique operands and operators than its peers is no more fault-proneness than they are.)

In the following, as Gyimothy *et al*. [10] and Basili *et al*. [11] did, we will test the null hypothesis for each metric on LAMOST and either accept the null hypothesis or reject it (in this case, we will accept the corresponding alternative hypothesis).

3. Analysis

In this section, we will assess empirically whether the metrics defined above are useful predictors of fault-prone functions. We will describe the analyses we performed to discover the relationships between the values of the metrics and the risk of the functions of the software system. The core of the system to be tested in this paper includes about 70 primary functions and about 8000 lines of codes. Table I provides common descriptive statistics of the metric distributions. The system is a medium system. In the test stage, we only tested 40 functions and used the results from the 40 functions to predict risks of the left functions. During the test, we not only calculated the bugs in each function, but also gave each bug a risk degree. We divided the risk degree into five levels: none, low, medium, high, higher. Then we can define a risk coefficient which is between 0 to 2 for each function by taking account of both the number of bugs and risk degree of each bug. In this experiment, we applied both univariate and multivariate analyses [10], [11]. Univariate analysis is used to examine the effect of each metric separately, while multivariate analysis examines the common effectiveness of all metrics. Firstly, we employed Bayesian belief networks, which was a statistical classification method. Then we chose the nearest neighbor method, rough set method and the decision tree method. Nearest neighbor method and rough set are able to predict the risk using just one metric and it is also possible to consider several metrics together for prediction. In decision tree method, which is more or less based on statistics, we only predicted the risk using all metrics, because we must construct a tree in this method, but one node tree cannot exert the advantages of the decision tree method for prediction.

In our experiment, we trained the system in two different ways as Gyimothy *et al*. [10] did. These are:

1. We considered only two categories: one category which included the function whose risk coefficient is equal or less than 1, and the other contained the left functions. We will use the <low, high> notation for this case.
2. We considered three categories [12]: one including the function which had no bugs, another with risk coefficient between 0 and 1or equal to1, and a category with risk coefficient greater than 1. We will use the <low, medium, high> notation for this case.

We compared the results of three methods (BBN, NN and RS) for individual metric and obtained similar results in all cases. Also, we compared the results of four methods (BBN, NN, RS and DT) for all metrics. We will present the results of the methods once at each time.

In our experiment, we applied the method of 5-fold cross-validated [10] for learning and testing. This means that we divided the training data into 5 equal parts and then performed the learning process five times. Each time, we chose another part for testing and used the remaining four parts for learning. Then, the average values were calculated from the five different testing results. We applied this procedure to the four classification approaches. At the same time, we choose three measures-accuracy, precision and F_measure to completely evaluate the results of the four methods. These measures are defined as

$$accruracy = \frac{TP}{pos+neg} \quad (1)$$

$$precision=\frac{TP}{TP+FP} \quad (2)$$

$$recall = \frac{TP}{TP+FN} \quad (3)$$

$$F\_measure = \frac{2*precision*recall}{precision+recall} \quad (4)$$

where *pos* is the number of positive samples, *neg* is the number of negative samples, *TP* is number of true positives and *FP* is the number of false positives.

Metrics	Accuracy	Precision	F_measure
CDENS	75.71%	0.60	0.67
CONTROL	74.29%	0.59	0.66
D	81.43%	0.84	0.81
V	78.57%	0.80	0.80
EXEC	81.43%	0.82	0.81
LOC	75.71%	0.64	0.69
N	78.57%	0.80	0.78
NSC	77.14%	0.79	0.77
NSTAT	81.43%	0.82	0.81
SLOC	72.86%	0.66	0.69
V(G)	75.71%	0.58	0.78
n	77.14%	0.80	0.75

Table 2. Result of using BBN for Individual Metrics (2 Categories)

3.1 Bayesian Belief Networks

Fenton *et al*. [1] used Bayesian belief networks for risk prediction and found that it had many advantages comparing with traditional methods which were regression-based. Bayesian belief networks are powerful tools for modeling causes and effects in a wide variety of domains. They specify joint conditional probability distributions. They are compact networks of probabilities that capture the probabilistic relationship between variables, as well as historical information about their relationships. These networks also

Metrics	Accuracy	Precision	F_measure
CDENS	57.14%	0.33	0.42
CONTROL	57.14%	0.33	0.42
D	58.57%	0.40	0.47
V	60.00%	0.46	0.50
EXEC	62.86%	0.52	0.53
LOC	58.57%	0.44	0.48
N	57.14%	0.36	0.43
NSC	57.14%	0.36	0.43
NSTAT	62.86%	0.48	0.52
SLOC	57.14%	0.42	0.46
V(G)	57.14%	0.33	0.42
n	58.57%	0.50	0.52

Table 3. Result of using BBN for Individual Metrics (3 categories)

offer consistent semantics for representing causes and effects (and likelihoods) via an intuitive graphical representation on which learning can be performed. A Bayesian belief-network is a directed acyclic graph (DAG) with a conditional probability distribution for each node. The DAG structure of such networks contains nodes representing domain variables, and arcs between nodes representing probabilistic dependencies. On constructing Bayesian networks from data sets, we use nodes to represent data attributes.

Table II presents the results of using Bayesian belief networks for individual metrics in two categories. It shows that D, EXEC and NSTAT have the same and largest accuracies (81.43 percent), and their precisions and F_measures are also much larger than others, especially D, which has the largest values (0.84 and 0.81 separately). The analyses above confirm that D is the best predictor for risk proneness. Although SLOC and CONTROL have accuracies more than 50 percent, they are much lower than 75 percent while those of others are higher than this value. Especially, precision and F_measure of CONTROL are the smallest. So SLOC and CONTROL can not be used as good predictors in this case. The results of other seven metrics are worse than D, EXEC and NSTAT, but better than SLOC and CONTROL.

Table III shows us the results of using Bayesian belief networks for individual metric in three categories. From Table III, we can find that EXEC has the largest accuracy, precision and F_measure values, so in this case, EXEC can be used as the best predictor. NSTAT is also good except that precision is a little lower. *n* has good precision and F_measure values, 0.50 and 0.52 separately, but its accuracy is a little lower. Therefore, n is a better predictor than other metrics except EXEC and NSTAT. From the two tables, it is easy to find that dividing the functions into two categories using these metrics independently by BBN is better than dividing them into three.

Table IV is the results of using Bayesian belief networks for all metrics including two and three categories. When all metrics are used to classify these functions, it is found that the results of dividing them into two categories are better than dividing into three. Although applying several metrics (such as D, EXEC, NSTAT) singly in Bayesian belief networks have higher accuracy, precision and F_measure values than applying all metrics in the case of two categories, it is simple to see that in the remaining situations using all metrics has a much better effect than using individual one.

Categories	Accuracy	Precision	F_measure
2	78.57%	0.80	0.80
3	62.86%	0.50	0.54

Table 4. Result of using BBN for ALL metrics

3.2 Nearest Neighbor

In this section, we will present our results of employing nearest neighbor method to predict the risk of a function with the help of metrics as predictors. Nearest neighbor classifiers are based on learning by analogy. They are comparatively simple and easy to do. Nearest neighbors classifiers are instance-based or lazy-learners in that they store all of the training samples and donot build a classifier until a new (unlabeled) sample needs to be classified. Instance based learning (also called memory-based learning) is a non-parametric inductive learning paradigm that stores training instances in a memory structure on which predictions of new instances are based. The approach assumes that reasoning is based on direct reuse of stored experiences rather than on the application of knowledge (such as models or decision trees) abstracted from experience. In our experiment, the most similarity between the new instance and a sample in memory indicatesthat the new instance belongs to the class of the sample. Table V and Table VI show the results of using only one metric by this method with two and three categories separately.

From Table V, we know that V has the largest accuracy and F_measure, although its precision is a little lower than that of D. Therefore, it must be the best predictor in this case. D is also better than other metrics, because it has the largest precision, and its accuracy and F_measure are the second highest separately. EXEC and N have the similar and better results (accuracy is 74.29 percent, precision is 0.77 and F_measure is 0.74). LOC is not as good as EXEC and N, but better than n as predictor. The accuracies, precisions and F_measures of SLOC, V(G), NSTAT and CDENS are lower than those of others, hence, they seem less useful.

Table VI is the results of using nearest neighbor for individual metric in three categories. From Table V, we can see that N has the largest accuracy, precision and F_measure. Therefore, we can conclude that N is the best predictor in this case. V has the second largest accuracy, precision and F_measure. So V is useful for prediction too. The accuracies of the CDENS, SLOC, V(G) and NSTAT are significantly smaller than the others, and also their precisions and F_measures are lower as they are in two categories in which

Metrics	Accuracy	Precision	F_measure
CDENS	70.00%	0.68	0.68
CONTROL	72.86%	0.75	0.71
D	74.29%	0.79	0.75
V	77.14%	0.78	0.78
EXEC	74.29%	0.77	0.74
LOC	74.29%	0.73	0.73
N	74.29%	0.77	0.74
NSC	72.86%	0.79	0.72
NSTAT	67.14%	0.67	0.67
SLOC	64.29%	0.67	0.65
V(G)	65.71%	0.68	0.65
n	74.29%	0.75	0.74

Table 5. Result of using NN for Individual Metrics (2 categories)

SLOC is the worst while CDENS in three categories. Therefore, CDENS, SLOC, V(G) and NSTAT are also less helpful in this prediction. V and N are the better predictors than others in this classification.

We also use all metrics together in this method to classify the functions into two categories and three categories respectively. Table VII presents the results. Using the twelve metrics together for prediction has the accuracy as high as V in two categories, but higher precision and lower F_measure than those of V. Therefore, when we need, we can choose all metrics together for prediction in this method. Classifying the functions into three categories is not as advisable as two categories. The values of using all metrics together in three categories are very low.

3.3 Rough Set

Rough set theory can be used for classification to discover structural relationships within imprecise or noisy data. It applies to discrete-valued attributes. Continuous-valued attributes must therefore be discredited prior to its use. Rough set theory is based on the establishment of equivalence classes within the given training data. All of the data samples forming an equivalence class are indiscernible, that is, the samples are identical with respect to the attributes describing the data. Given real-world data, it is common that some classes cannot be distinguished in terms of the available attributes. Rough sets can be used to approximately or "roughly" define such classes. A rough set definition for a given class is approximated by two sets- a lower approximation of *C*. The Lower approximation of *C* consists of all of the data samples that, based on the knowledge of the attributes, are certain to belong to *C* without ambiguity. The upper approximation of *C* consists of all of the samples based on the knowledge of the attributes, cannot be described as not belonging to *C*. The lower and upper approximations for a class C are shown in Figure 1, where each rectangular region represents an equivalence class. Decision rules can be generated for each class. Typically, a decision tables is used to represent the rules.

Metrics	Accuracy	Precision	F_measure
CDENS	37.14%	0.43	0.37
CONTROL	52.86%	0.53	0.50
D	50.00%	0.53	0.49
V	57.14%	0.58	0.56
EXEC	50.00%	0.50	0.49
LOC	57.14%	0.49	0.55
N	60.00%	0.63	0.60
NSC	54.29%	0.61	0.51
NSTAT	48.57%	0.42	0.45
SLOC	44.29%	0.44	0.41
V(G)	45.71%	0.49	0.43
n	55.71%	0.49	0.47

Table 6. Result of using NN for Individual Metrics (3 categories)

Categories	Accuracy	Precision	F_measure
2	77.14%	0.80	0.76
3	48.57%	0.51	0.47

Table 7. Result of using NN for ALL Metrics

Table VIII gives the results for the two categories classification using only one metric. As can be seen, D has the highest values (72.85 present, 0.71 and 0.67) , once again confirming that D is the best predictor. EXEC and V are a little worse than D, but better than other metrics, which indicates EXEC and V are not bad predictors. The results of LOC are also not bad, only lower than D, EXEC and V. the values of CDENS and n are more or less equal and these values are acceptable, only n's F_measure higher than that of CDENS. Similar with CDENS and n, N and NSC have the same accuracy (58.57 percent) which is not high. The other values of NSC are better than those of N's. In this case, V(G) has the worst accuracy (51.43%) and the other values are not high (0.55 and 0.51 for precision and F_measure respectively), so it is a poorer predictor in this case. CONTROL and NSTAT are only a little better than V(G). Therefore, they are also not good predictors.

We also performed the experiment, using rough set to classify the functions into three categories (see Table IX). It is clear that classifying functions into two categories are better than three as same as the BBN and NN. D, again, is the best predictor with the highest values. V is just a little poorer than D, but better than the remained metrics. NSTAT, N, SLOC, LOC and CONTROL all have the accuracy more than 50 percent and their precision and F_measure are not high. NSC and CDENS have the similar results. The accuracies of

Metrics	Accuracy	Precision	F_measure
CDENS	60.00%	0.52	0.50
CONTROL	57.14%	0.59	0.57
D	72.85%	0.71	0.67
V	70.00%	0.67	0.64
EXEC	71.43%	0.61	0.60
LOC	61.43%	0.54	0.51
N	58.57%	0.59	0.58
NSC	58.57%	0.64	0.59
NSTAT	52.86%	0.53	0.52
SLOC	58.71%	0.63	0.53
V(G)	51.43%	0.55	0.51
n	60.00%	0.52	0.51

Table 8. Result of using RS for individual metrics (2 metrics)

Metrics	Accuracy	Precision	F_measure
CDENS	50.00%	0.33	0.28
CONTROL	51.43%	0.34	0.28
D	58.57%	0.49	0.32
V	57.14%	0.37	0.31
EXEC	48.57%	0.30	0.28
LOC	52.85%	0.30	0.26
N	54.29%	0.42	0.35
NSC	50.00%	0.33	0.30
NSTAT	55.71%	0.41	0.32
SLOC	52.86%	0.36	0.33
V(G)	48.57%	0.31	0.30
n	48.57%	0.45	0.37

Table 9. Result of using RS for individual metrics (3 metrics)

Categories	Accuracy	Precision	F_measure
2	61.57%	0.60	0.61
3	41.43%	0.33	0.29

Table 10. Result of using RS for ALL metrics

EXEC, V(G) and n are lower than 50 percent, only 48.57 percent, so they are not effective in this classification.

Table X provides us the results of using rough set to do multivariate analysis. The values of two categories are much better than those of three. When we classify the functions into two categories using rough set, some univariate accuracies (D, V and EXEC) are better than that of mutivariate while only the precisions and F_measures of D and V are higher that that of multivariate. In the case of three categories, the results of multivariate experiment are poorer than those of univariate. Therefore, classifying the functions into three categories is not as advisable as two categories using rough set.

3.4 Decision Tree

In this section, we employed decision tree to predict the risk of a function with the help of all metrics as predictors. Decision tree (DT) is one of the most popular and inductive learning algorithms. It is like a tree structure of flow chart. The nodes inside of the tree correspond to the test of an attribute, and the links (linking attribute values and the leaves) present the output of a test, while the leaves represent the classes or the classification of the classes. To induce a DT, the most important attribute (according to an attribute selection criteria, such as information gain, GainRatio, etc.) is selected and placed at the root; one branch is made for each possible attribute value. This divides the samples into subsets, one for each possible attribute value. The process is repeated recursively for each subset until all samples at a node have the same classification, in which case a leaf node is created. To classify a sample we start at the root node of the tree and follow the path corresponding to the sample's values until a leaf node is reached and the classification is obtained. To prevent overtraining DT is typically pruned. The most popular DT learning algorithm is Quinlan's C4.5 [6]. We have used the Weka [5] implementation of the C4.5 algorithm in our experiments with information gain as the attribute selection criterion.

Suppose that there are a total of *m* classes denoted by $C=\{C_1, C_2,\ldots, C_m\}$, at a particular node in the tree, and there be *N* training samples represented by,

$(a(1), b(1),\ldots; t(1)), (a(2), b(2),\ldots; t(2)),\ldots, (a(N), b(N),\ldots; t(N))$

where, *a*(i), *b*(i),…are vectors of *n* attributes and *t*(*i*)"*C* is the class label. Of the *N* examples, N_{C_k} belong to class C_k. The decision rule at the node splits these examples into *V* partitions, or *V* child nodes, each of which has $N^{(v)}$ samples. In a particular classification, the number of samples of class C_k is denotes by $N_{C_k}^{(v)}$. The information gain resulting from splitting the rules bases on attribute *A* can be written as,

$$Gain\ (A) = [\sum_{k=1}^{m} -(\frac{N_{C_k}}{N})\log(\frac{N_{C_k}}{N})] - [\sum_{v=1}^{V}(\frac{N^{(v)}}{N})\sum_{k=1}^{m} -(\frac{N_{C_k}^{(v)}}{N^{(v)}})\log(\frac{N_{C_k}^{(v)}}{N^{(v)}})] \quad (5)$$

In the experiment of DT, when we assign m=2, the result is about two categories, while m=3 about three categories.

Table XI presents the results of using decision tree for all metrics in two and three categories. As can be seen, DT for dividing the functions into two categories is better and its accuracy is 72.86 percent in the first classification (<low, high>), 52.86 percent for the second. From the precision and F_measure, we can see that DT can be used for prediction effectively.

Categories	Accuracy	Precision	F_ measure
2	72.86%	0.75	0.72
3	52.86%	0.48	0.47

Table 11. Result of using DT for ALL Metrics

Comparing Table XI with Table X , TableVII and Table IV, it can be found that for LAMOST using DT is not the better choice (because its accuracy is lower than that of BBN and NN, its 72.86 percent while BBN's and NN's 78.57 percent and 77.14 percent separately).

3.5. Discussion and the Validation of the Hypotheses

In this section, we will study the results described in the previous sections to validate our hypotheses. We summarized the descriptive values (only 2 categories) of the three models in Table XII. Because dividing the functions into two categories is much better than into 3 categories, we only

Metrics	Model	Accuracy	Precision	F_measure
CDENS	BBN	75.71%	0.60	0.67
	NN	70.00%	0.68	0.68
	RS	60.00%	0.52	0.50
CONTROL	BBN	74.29%	0.59	0.66
	NN	72.86%	0.75	0.71
	RS	57.14%	0.59	0.57
D	BBN	81.43%	0.84	0.81
	NN	74.29%	0.80	0.75
	RS	72.85%	0.71	0.67
V	BBN	78.57%	0.80	0.80
	NN	77.14%	0.78	0.78
	RS	70.00%	0.67	0.64
EXEC	BBN	81.43%	0.82	0.81
	NN	74.29%	0.77	0.74
	RS	71.43%	0.61	0.60
LOC	BBN	75.71%	0.64	0.69
	NN	74.29%	0.74	0.73
	RS	61.43%	0.54	0.51
N	BBN	78.57%	0.80	0.78
	NN	74.29%	0.77	0.74
	RS	58.57%	0.59	0.58
NSC	BBN	77.14%	0.78	0.77
	NN	72.86%	0.79	0.72
	RS	58.57%	0.64	0.59
NSTAT	BBN	81.43%	0.82	0.81
	NN	67.14%5	0.67	0.67
	RS	2.86%	0.53	0.52
SLOC	BBN	72.86%	0.66	0.69
	NN	64.29%	0.67	0.65
	RS	58.71%	0.63	0.53
V(G)	BBN	75.71%	0.58	0.65
	NN	65.71%	0.68	0.65
	RS	51.43%	0.55	0.51
n	BBN	77.14%	0.74	0.75
	NN	74.29%	0.80	0.74
	RS	60.00%	0.52	0.51
All	BBN	78.57%	0.80	0.78
All	NN	77.14%	0.80	0.76
All	DT	72.86%	0.75	0.72
All	RS	61.57%	0.60	0.61

Table 12. Summary of Accuracy, Precision and F_measure

consider the 2 categories. From Table XII we can see that all accuracies of the metrics are more than 50 percent, so the results of classification can be accepted. Using the numbers in TableXII and the initial data of samples after statistical study, we can check the hypotheses.

CDENS hypothesis: In all methods of BBN, NN and RS, CDENS's accuracies, precision and F_measure are a little lower than most metrics including using all metrics except that CONTROL, N, NSC, NSTAT, SLOC and V(G). The results of N and NSC are only lower than those of CDENS in RS. SLOC and V(G) have a worse effect than CDENS, while CONTROL and NSTAT are worse in two of the three methods. So although CDENS is effective for risk prediction, it is not good enough if take as a predictor. With the related numbers and initial data, we rejected the null hypothesis of CDENS and accept the alternative hypothesis.

CONTROL hypothesis: CONTROL has the same situation with CDENS. Its accuracy, precision and F_measure in BBN and RS are a little lower than those of CDENS while higher in NN, so we can conclude that CONTROL is also not a good enough predictor. Similar to CDENS, we rejected the null hypothesis of CONTROL and accepted the alternative hypothesis.

D hypothesis: The results of D in all methods are significant, especially having the highest accuracies (81.43 percent and 72.58 percent), precisions (0.84 and 0.71) and F_measures (0.81 and 0.67) in BBN and RS respectively. Although in NN its accuracy and F_measure are not as high as V, it has the largest precision. We can say that D is the best predictor. Taking into account these results and initial data, we rejected the null hypothesis of D and accepted the alternative hypothesis.

V hypothesis: V also has the good results with high accuracy, precision and F_measure in all three methods. Although its results are a little lower than D's in BBN and RS , its accuracy and F_measure are higher than those of D in NN. So we can see that it is also a good predictor. With the related numbers and initial data, we rejected the null hypothesis of V and accepted the alternative hypothesis.

EXEC hypothesis: The results of EXEC are like those of D. The little differences between them are their precisions and F_measures, specifically, those of D are higher in all methods. Although EXEC may not as good as D in risk prediction, we also can conclude that EXEC is an effective predictor. After analyzing these numbers and initial data, we rejected the null hypothesis of EXEC and accepted the alternative hypothesis.

LOC hypothesis: The precisions and F_measures in BBN and RS are a little smaller, especially in RS (only 0.54 and 0.51), and the accuracy is not significantly high in all methods, so it is not a good enough predictor. Considering the results and the initial data, we rejected the null hypothesis and accepted the alternative hypothesis.

N hypothesis: N has the same accuracy (78.57 percent) and precision (0.80) with V in BBN while F_measure is a little smaller. But in NN and RS, N has a little lower result than V. So we can make the conclusion that N is not a bad predictor. With the results and the initial data, we rejected the null hypothesis of N and accepted alternative hypothesis.

NSC hypothesis: The results of NSC are more or less equal to those of N, so like N, NSC is not a bad predictor. After studying the values and initial data we rejected the null hypothesis and accepted the alternative.

NSTAT hypothesis: The results of NSTAT by using BBN are good (accuracy 81.43 percent, precision 0.82 and F_measure 0.81, better not only than those of most metric, but also than multivariate model), but by using NN and RS are poor (accuracy 67.14 percent and 52.86 percent, precision 0.67 and 0.53, F_measure 0.67 and 0.52 respectively in two methods). Therefore, it is difficult to decide whether NSTAT is a good predictor. We do not know the reason for these big differences, so it needs further investigation. Overall, though, we rejected the null hypothesis of NSTAT and accepted alternative hypothesis after analysis.

SLOC hypothesis: The accuracies (72.86 percent in BBN and 64.29 percent in NN) of SLOC are the lowest in all the metrics in both methods. Its two precisions and F_measures are also quite low, the highest 0.69. And its results in RS is also not good. Therefore we can conclude that SLOC is not a good predictor. We also analyzed initial samples, and we decided to reject the null hypothesis and accept the alternative.

V(G) hypothesis: The results of V(G) is worse than SLOC. Its accuracy, precision and F_measure in NN are the second lowest, only better than SLOC and are the worst in RS. So we also can say that V(G) is not a good predictor. Considering the values and the initial data, we decided to reject the null hypothesis of V(G) and accept the alternative hypothesis.

n hypothesis: The results of n are not bad except its precision and F_measure are a little lower in RS. Its results are opposite to that of NSC in BBN and NN. In BBN, n has a little higher values than NSC, but in NN, NSC is better than n. All in all, n has the similar situation with NSC and they are both not good enough predictors. Through analysis, we rejected the null hypothesis and accepted the alternative.

We also compare the results of the multivariate analyses to each other. BBN model has slightly better values than those obtained from nearest neighbor, rough set and decision tree. The rough set model produced the weakest model with lowest accuracy, precision and F_measure. We can see in this kind of software, BBN has the better effect than NN and RS when using one metrics for classification.

4. Conclusion and future work

Our main observations are following:

1. The three assessment methods (BBN, NN and RS) for individual metric employed yield similar results.
2. The D metric seems to be the best in predicting the risk of the functions.
3. The EXEC and V metrics performed fairly well, and because they can be easily calculated, they seem to be suitable for quick prediction.
4. N has good values, although worse than D, EXEC and V, better than the other metrics. And it is also easy to be calculated.
5. The results of NSTAT in the three methods (BBN, NN and RS) have significantly differences, so it needs more investigation.
6. CDENS, CONTROL, LOC, V(G) and SLOC are all not good risk predictors.
7. NSC and n are not good enough to be quality predictors.
8. In this case, BBN is the best method for classification no matter whether we use one metric individually or all together.
9. When using univariate metric to do analysis, RS performed the worst model.
10. When we use all metrics for risk prediction, RS has the worst effect.

The results of our models are not yet satisfactory, so we have to analyze the models and examine whether other metrics can improve them. We will consider some combination of selected metrics for prediction and find which combination is more effective. We will also check whether

multiple models perform better when combined in some way.

References

[1] Fenton, N., Neil, M (1999). Software Metrics and Risk, *Proc. Second European Software Measurement Conference* (FESMA 99), p. 39-55, October.

[2] Heckerman, D., Geiger, D., Chickering, D.M (1994). Learning Bayesian Networks: the Combination of Knowledge and Statistical Data, *Technical Report MSR-TR-94-09*, Microsoft Research.

[3] Cheng, J., Bell, D.A., Liu, W (1997). An Algorithm for Bayesian Belief Network Construction from Data, *Proc. AT @ATAT*, p.83-90.

[4] Han, J., Kamber, M (2001). *Data Mining – Concepts and Techniques*. Morgan Kaufman Publisher, p. 314–315.

[5] Witten, I., Frank, E (2000). *Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann.

[6] Quinlan, J.R (1993). *C*4.5: *Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

[7] The Krakatau Professional Homepage http://www.powersoftware.com/kp/

[8] Rosenbergÿ. L., Hammer, T., Shaw, J (1998). Software Metrics and Reliability, *Proc. Ninth International Symposium on Software Reliability Engineering* (ISSRE 98).

[9] Guo, P., Lyu, M. R (2000). Software Quality Prediction Using Mixture Models with EM Algorithm. Proc. First Asia-Pacific Conference on Quality Software (APAQS 2000), p.69-78.

[10] Gyimothy, T., Ferenc, R., Siket, I (2005). Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction, *IEEE Trans. Software Eng.*, 31 (10) 897-910.

[11] Basili, V.R., .Briand, L.c., Melo, W.L (1996). A Validation of Object-Oriented Design Metrics as Quality Indicators, I*EEE Trans. Software Eng*, 22 (10) p.751-761.

[12] Cherukuri, R (2003). An Empirical Study of a Three-Group Software Quality Classification Model, M.S. thesis, Florida Atlantic Univ., Florida, USA.

[13] Pedrycz, W., Succi, G., Chun, M.G., (2002). Association Analysis of Software Measures, *Int. J. of Software Engineering and Knowledge Engineering*, 12 (3) p.291-316.

[14] Pedrycz, W., Succi, G., Musilek, P., X. Bai, X (2001). Using Self-organizing Maps to Analyze Object Oriented Software Measures, *J. of Systems and Software*, 59, 65-82.

[15] Xing, F., Guo, P., Lyu, M.R (2005). A novel method for early software quality prediction based on support vector machine", *Proc. of 16th International Symposium on Software Reliability Engineering,* (ISSRE'2005) p. 213-222.

Dr Ping Guo is currently a Professor with the School of Computer Science at Beijing Institute of Technology, China and Computer Science Department, Beijing Normal University, China. From 1993 to 1994, he was with the Department of Computer Science and Engineering, Wright State University, Dayton, USA as a visiting faculty. His current research interests include computational intelligence, image processing, software reliability engineering, optical computing, and spectra analysis.