Comparing Fault-Proneness Estimation Models

P. Bellini, I. Bruno, P. Nesi, D. Rogai
Department of Systems and Informatics, University of Florence
Via di S. Marta 3, 50139, Florence, Italy
http://www.dsi.unifi.it/~nesi, nesi@dsi.unifi.it, tel: +39-055-4796523, fax: +39-055-4796363

Abstract

Over the last years, software quality has become one of the most important requirement in the development of systems. Fault-proneness estimation could play a key role in quality control of software products. In this area, much effort has been spent in defining metrics and identifying models for system assessment. Using these metrics to assess which parts of the system are more fault-proneness is of primary importance. This paper reports a research study begun with the analysis of more than 100 metrics and aimed at producing suitable models for fault-proneness estimation and prediction of software modules/files. The objective has been to find a compromise between the fault-proneness estimation rate and the size of the estimation model in terms of number of metrics used in the model itself. To this end, two different methodologies have been used, compared, and some synergies exploited. The methodologies were the logistic regression and the discriminant analyses. The corresponding models produced for fault-proneness estimation and prediction have been based on metrics addressing different aspects of computer programming. The comparison has produced satisfactory results in terms of fault-proneness prediction. The produced models have been cross validated by using data sets derived from source codes provided by two application scenarios.

Index terms: maintenance, fault-proneness estimation, fault-proneness prediction, cross validation, empirical validation.

1. Introduction

Software reliability is one of the most important features for many software applications. In many cases, it is considered the most relevant quality indicator of software modules. A particular attention to this feature is given by companies that develop communication systems (embedded with real-time constraints) and medical applications. In both cases, a high reliability has to be guaranteed to limit damages and high maintenance costs. High reliability frequently implies high investments for software testing, verification and validation. These activities are very expensive to be exhaustively performed; on the other hand, their costs can be reduced thanks to methods and models for estimating the fault-proneness of system modules, thus allowing the identification of the most critical components that have to be more carefully verified [11]. The details about software model verification and validation and improvement are not taken into account in this paper. A software module is stated to be fault-prone when there is a high risk that a fault will be discovered during its operative work [11]. Typically when a software module has been identified as fault-prone, it can be verified and tested more accurately. This increases the probability of identifying future bugs and reduces the risks of faults and thus the maintenance costs.

In order to estimate/predict the fault-proneness of software modules several different technologies and models have been proposed, in the literature. They can be classified into several groups: Logistic Regressions [1], [7], [10], Discriminant Analysis [12], [15], Discriminant power [16], Discriminant Coordinates [13], Optimized reduction sets [2], Neural Networks

[9], fuzzy Classification [5], Classification and Regression Trees [11].

Most of these solutions, the aim is to propose a technique, capable to create a model for estimating the fault-proneness of software modules or classes (in the event of Object-Oriented programming languages). The models are typically based on set of technical low-level metrics that can be estimated by processing the source code. In many cases, the metrics of the model are selected from a larger set of possible metrics according to the model definition process itself. The reduction in the number of metrics allows obtaining a model that can be used to obtain easier and faster fault-proneness estimations. In addition, the identification of a restricted number of metrics allows also to identifying better the technical aspects related to the fault-proneness of a given software module in a given context or software type. This identification allows providing some guidelines for a more careful verification/validation of software modules; for example, the metric which was more relevant in estimating the fault-proneness of a source can denote which kind of additional test should be performed on such source.

Once the model is obtained, it has to be assessed and validated in order to confirm the confidence of the obtained results. Frequently, the validation is performed by using empirical and statistical significance of coefficients to verify the confidence of the numbers that are the model parameters. Other types of approaches are based on cross-validation; a part of the data set (a data subset) is used to create the model (typically for training or model definition) and the other subset to perform the validation.

As it has been highlighted in many research works cited above, only a few metrics are relevant to obtain a good fault-proneness estimation/prediction of software modules. An investigation on how all metrics are related to fault-proneness estimation and how many of these metrics are needed to obtain a good model has been performed in many papers. In many cases, the selected metrics covered different areas/features of software systems and the capability of a model with respect to the others is based on the combination of the mathematical model and the metrics which produce estimation. In fact, in some cases, it occurs that the metrics producing high quality results with a model are not capable to provoke similar rates when they are used in other models. The large number of metrics selected as independent variables and the several aspects taken into account by previous works, suggested that models including metrics that consider different aspects can be suitably used for defining model for fault proneness estimation/prediction. This assumption has been also partially supported in [3], where the model obtained relevant results. This paper confirms that a model comprised of metrics that take into account several aspects (coupling and cohesion, structural, functional) can be suitably applied to faulty class identification obtaining a very high value of confidence. The experiments reported in this paper were based on the data sets briefly described in the following.

The main goal of the study reported in this paper has been to define and validate models for fault-proneness estimation. The planned goal was mainly to define a model independent from a specific software context with fault-proneness estimation greater than the 90% (the meaning of this number will become clearer later in the paper), limiting the number of metrics used in the model to a manageable number, that could be less than 10-15 metrics.

For this purpose, two different methodologies for the production of fault-proneness models have been considered: statistical analysis based on logistic regression; discriminant analysis based on risk-threshold. The starting point was to collect the measures of several metrics available in the literature, for the modules/files of two real industrial software systems. The first goal of the performed analysis and comparison of the different methodologies has been to verify if the two models lead to extract the common metrics from the larger initial set of metrics. A part of the work was aimed at identifying a model for fault-proneness estimation independent on the project context: company, methodology, software tool domain and development people, etc. It has been also interesting to compare the results which can be obtained by statistical and discriminant analysis in order to verify which of them can be better ranked in fault-proneness estimation, on the basis of the same data sets and metrics.

This paper is organized as follows. In Section 2, the statistical technique based on principal components and logistic regression is described. In Section 3, the discriminant analysis on risk-threshold is explained.

Section 4 reports the description of the data sets used for the empirical validation and for the cross validation of the proposed models for fault-proneness estimation. In Section 5, the experimental results are reported comparing: statistical analysis and discriminant analysis models. Conclusions are drawn in Section 6.

2 Statistical Techniques for Data Analysis

Several statistical techniques have been applied for defining software fault-proneness models (see [1], [7], [10]). To this end two commonly used statistical techniques have been adopted. The former is the principal component analysis, while the latter is the multivariate logistic regression.

The Principal Component Analysis (PCA) [4] is typically used to extract Principal Components (PCs) from the variance of a set of variables. Principal Components (PCs) are the orthogonal linear combinations of variables whose variance is equal to those observed. The first PC is the linear combination of variables which explain the maximum amount of variance in the data set. The other PCs are orthogonal to the previously extracted components and explain in turn the maximum amount of the residual variance. Usually only a small subset of all the variables has a large coefficient (loading) in a PC, and therefore only these variables should be considered as significant from a statistical point of view. The variables having a high loading usually identify the dimension captured, even if a certain degree of interpretation is required. In order to simplify the structure of the PCs, a rotation of the components is usually performed. This operation tends to reduce the loading of the coefficients that were small in the component matrix and to increase the loading of the already significant components. In this paper, the Varimax rotation method has been adopted [8].

3 Discriminant Analysis on Risk-threshold

Discriminant Analysis [12], has been used in software fault classification and estimation and divided into some categories: Discriminant power [16], Discriminant Coordinates [13], Optimized reduction sets [2].

In this paper, a discriminant analysis based on risk has been presented [15]. The starting point for the risk analysis was the definition and measurement of a set of experimental parameters connected to the structure of software products in terms of code.

The number of faults recorded during the execution of a code source has been compared to certain critical values found in the same code for some of the measured parameters (which can be directly or indirectly estimated in terms of code metrics). In this way, the reliability level of a program is related to some metrics. In general, the aim of the metric is to identify those parameters which, in the light of the values measured, explain the presence of faults in software modules and which can be defined as risky metrics for that reason. The identification of these risky metrics and the files/modules which have a high risk to contain faults can be used to pre-process the modules during the releasing and testing session. The choice regarding which metrics have to be taken into account depends primarily on the application, not to mention a whole range of characteristics related to the programming environment, such as the language used, the available tools, the rules governing software development and internal testing, the types of problems faced, etc. Starting from a database of programs and signaled faults, the correlation between a set of metrics and the faults found in each module/file have to be modeled. After defining the model, this can be easily used in analyzing all new files/modules before testing session. The analytical steps of the model definition are:

1. The process starts with a large set of m metrics.
2. Each metric, i, has been measured in each file, j, as x_{ij} ($1 \leq i \leq m$, $1 \leq j \leq n$, being n the number of files/modules);
3. For each metric i, the mean value M_i and the standard deviation estimated on values obtained for all files/modules has been computed as $S_i = \sqrt{\frac{\sum_{j=1}^{n}(x_{ij}-M_i)^2}{n}}$, where $M_i = \frac{1}{n}\sum_{j=1}^{n}(x_{ij})$;
4. For each metric i, for each file j, the values S_{ij} have been considered as the offset of the metric evaluated on the j^{th} file from the metric mean value M_i, normalized on the standard deviation of the metric S_i: $S_{ij} = \frac{|x_{ij}-M_i|}{S_i}$;
5. The risk level of metric i, MRL_i, is calculated as $\sum_{j=1}^{n} R_j \cdot S_{ij}$ being R_j considered 1 if the file

j has reported faults, 0 else. All the risk values MRL_i have been normalized with respect to the sum of all MRL.

6. The risk level of file j, FRL_j, is computed on the basis of the metric value and their risk level as: $FRL_j = \sum_{i=1}^{m} MRL_i \cdot x_{ij}$; all the risk levels FRL_j have been normalized with respect to the sum of all FRL.

The Discriminant Analysis on risk-threshold is performed by splitting the files/modules data set into two data subsets for training and validation. The objective of the training phase is to compute the risk level for each metric MRL_i on the basis of the processed files/modules. On the basis of the MRL_i, the risk level of a source file/module FRL_j is computed.

The risk-threshold, which is responsible of the file/module classification, has been taken as in the middle of the average values of FRLs of the two disjoint classes: modules with faults and those without. The risk-threshold is defined as $(MFRL_f + MFRL_{nf})/2$ where $MFRL_f = avg(FRL_j)_{j \in fault}$ and $MFRL_{nf} = avg(FRL_j)_{j \in nonfault}$ in the middle between the two mean values extracted from each class. An uncertain interval is arranged around the risk-threshold (not all the modules could be classified).

The modules of the validation subset have their own risk level FRL. Such FRL has been compared to the risk-threshold in order to classify the file/module. Since the uncertain interval is arbitrary and our objective was to produce a model with an acceptable fault-proneness estimation rate, the risk-threshold uncertain interval was set to obtain at least 90% of general classification.

It is evident that the set of metrics have to be reduced to the minimum set of representative parameters, to avoid metrics which are not useful in the analysis or which introduce noise in the evaluation of data. Empirical methods to reduce the metric set have been applied as described in the Section 5.2.

4 Experiment data sets description

The experiments to produce the fault-proneness models have been carried out processing sources from two different software projects. They present files coming from different environments of development and application fields, both of them can be classified as small/medium systems written in C language. Their main properties are:

- INDUSTRIAL, industrial low-level application context, 235 modules to implement an embedded network router, with and without cryptographic features.

- MEDICAL, graphical user interface and data access context, 280 modules to implement front-end application of management software for medical purposes.

The experimental data set have been collected with heterogeneous source types in order to highlight similarities and differences among development scenarios. The companies which are the owners of these software systems have not provided the needed authorization to mention them. The considered metrics to create the data sets have been estimated by using two assessment tools for the analysis of programming code. These measuring tools are: CPP-Analyzer (see [17]) and PAMPA (Project Attribute Monitoring and Prediction Associate) [14]. These tools estimate the metrics defined by the corresponding research teams, and also several other metrics published in the literature, for a total of more than 100 different metrics at system module level for C language code.

The data related to faults have been used as the dependent variable for the following study. In particular, a software module has been considered faulty if at least a fault has been recorded. Since the number of cases, against whose the model could be fitted was limited, no further inspection on the relationships between faulty modules with 2 or more faults and metrics has been carried out.

The cited tools can process C source codes to produce measures of more than 100 metrics. All these metrics could be potentially considered as independent variables for the definition of models for estimation and prediction of fault-proneness in software systems. Details about these metric suites can be recovered in the above mentioned papers.

The considered metrics cover all the aspects of software measures presented in the quoted literature. Due to lack of space, it is impossible to describe such metrics and quote all the corresponding technical papers in which they are defined and discussed.

5 Experimental Results

In this section, a short description of the procedures adopted for the data analysis and for the model definition are reported. The objective has been to find a model with a reasonable number of metrics needed to estimate fault-proneness in software. The first aim was to find a model capable to maintains acceptable computational costs (low number of metrics), while providing a prediction rate over 80% of the effective result. The main idea was to start from the analysis of the whole set of metrics so as to identify the most important metrics, on the basis of their contribution in estimating fault-proneness. To this end, at the beginning, all the metrics estimable by the CPP-analyzer and PAMPA tools have been considered. In this phase, no limitation on the number of metrics was adopted. For this reason, the 113 metrics calculated by the two adopted tools have been analyzed with descriptive statistic techniques. On the basis of descriptive statistic, meaningless variables with a null variance were removed.

Experiments were performed by using the above mentioned models and data sets mentioned before, and can be summarized as follows:

Statistical analysis

1. PCA for metric reduction has been applied selecting the metrics which are the most related to the Principal Components extracted (on INDUSTRIAL and MEDICAL data sets);
2. Logistic regression with different techniques has been used to find an early model;
3. PCA has been applied a second time for a further reduction;
4. Logistic regression has been performed to obtain a model with a smaller set of metrics;
5. The models identified by statistical analysis have been evaluated.

Discriminant analysis

1. Each data set (MEDICAL, INDUSTRIAL) has been separated into two sets called training and validation;
2. The risk-coefficient has been calculated for each file/module of the training set and mean values have been computed for fault and no-fault subgroups;
3. The risk-threshold has been fixed between the two means with a "neutral" range centered in the threshold value to exclude critical value from the classification;
4. The metric set has been clustered, grouping metrics of the same nature: complexity, size, etc.
5. The discriminant analysis has been performed on metric clusters and results have been produced;
6. The metric clusters have been empirically reduced on the basis of risk contribution in order to produce a reduced number of metrics with a reasonable fitting according to the above mentioned target results (e.g., 80%);
7. The models identified by discriminant analysis have been evaluated.

In the reported experiments, different approaches for creating fault-proneness models have been applied by using the same set of metrics. In the experiments, the metrics have been considerer as the main subject of the analysis, searching for the minimum set of them which could produce an acceptable rate (80%) for fault-proneness estimation.

5.1 Statistical analysis results

In this section, the statistical analysis has been applied in these two different data sets: INDUSTRIAL and MEDICAL.

Results on INDUSTRIAL data set

Before the application of the logistic regression on the INDUSTRIAL data set, the statistical method began with the PCA to reduce the set of variables to a smaller set. In this case, 8 principal components were considered. In fact, the component analysis process was stopped when the eigenvalue of the covariance matrix reached the value of 1.5 regardless of the variance, covering more than 78%. Over the 9^{th} principal component, the variance contribution did not increase anymore with respect to what was obtained

with the first 8 PCs. On the basis of the first 8 principal components, the most relevant metrics have been selected and the whole collection of metrics has been reduced to 58: the selection has been made on the basis of the contribution of each metrics to the PCs. To this end, the metrics with contribution outside a range of $[-0.7; +0.7]$ have been excluded, because of the low influence in the covariance matrix. The metrics related to the first principal component cover more than 50% of the variance.

After this preliminary selection, the logistic regression has been applied to identify the model according to this reduced set of metrics belonging only to the first principal component. Different techniques of logistic regression have been used on this metric set and data. The logistic regression with backward technique has reduced the metric set to only 18 metrics, while losing only a small percentage on the prediction rate.

Another model has been produced by using the whole set of 58 metrics concerning the 8 PCs, using conventional and backward regressions. In this case, better results were obtained, and the backward technique has been capable to reduce the model of 11 metrics by obtaining the prediction rate close to 90% with 47 metrics.

The statistical analysis on the basis of 47 metrics has produced better results, while still using a large amount of metrics. A further work has been performed to reduce the number of metrics. A new set of metrics has been extracted from the 6 PCs. From the first 6 PCs, 24 metrics have been selected considering the contribution parameters in the $[-0.85; +0.85]$ range. The conventional technique of logistic regression on the metrics related to the first component obtained a rate greater than 80%. On the whole set of metrics related to the 6 principal components the conventional logistic regression has not produced satisfactory results. In fact, the same prediction rate has been obtained with the backward technique which reduced the metric set from 24 to 17 metrics. In this case, the results have reached the 80% of fault-proneness estimation.

This last model based on 17 metrics with a rate of 80% has been more carefully analyzed. The metrics have been considered one by one, excluding some of them due to their conceptual irrelevance with the fault-proneness estimation problem. For example, a metric related to the number of macro definitions was eliminated. On the basis of this analysis, the metric set has been empirically reduced. The remaining 11 metrics maintained a prediction rate which was still on the 80% (the same achieved with 17 metrics set) as shown in Tab. 1.

backward - $threshold = 0.5$
+ empyrical reduction

	Actual		
Model	no fault	fault	% correct
no fault	144	30	82.76%
fault	16	45	73.77%
		total	80.43%

Table 1. Results of logistic regression with 11 metrics on all PCs of the second reduction on the INDUSTRIAL data set.

The obtained model has been evaluated considering the typical statistical coefficients of dichotomous classification. The values describing this model based on logistic regression are reported in Tab.2, where it is evident that the value of the Wald coefficients confirm the relevance of the metrics, except for the offset.

Results on MEDICAL data set

On the MEDICAL data set the first step of the principal component analysis has found 10 principal components which covered more than the 80% of the total variance. Considering the range of $[-0.7; +0.7]$, 55 metrics of the initial 113 have been selected for the 10 PCs. The logistic regression with backward technique reduced the number of metrics to 24: the prediction model for this set was near 90% .

An additional PCA on the 24 metrics identified that most of the variance was modeled by the first 3 principal components. Using the logistic regression with step-wise forward techniques only 5 metrics have been selected and the results were comparable with those obtained from whole set. On the other hand, the backward technique produced higher percentages including 14 metrics. In Tab. 3, more details about the model based on 5 metrics are reported. Please note that, in this case, all the metrics involved in the model are significant in terms of p-value and Wald. The metrics which have been selected by both models are NDR and DATADECLS, which are metrics that

Statistical coeff.	result	theoretic
Bravais-Pearson	0.531	1
Sensibility	0.600	1
Correctness	0.900	1
Completeness	0.737	1
Classification Corr.	0.804	1
Type I Errors	0.068	0
Type II Errors	0.127	0

Metric	β	Std. Error	Wald	p-value
CDF	-1.734	0.682	6.458	0.011
DATADECLS	-0.348	0.015	4.285	0.038
NAIV	-0.079	0.023	11.632	0.006
NCFB	0.023	0.008	7.053	0.003
NDDTU	0.042	0.013	9.767	0.001
NDR	0.072	0.020	12.337	0.004
NDS	-0.261	0.090	8.287	0.004
NDSTT	1.121	0.788	2.025	0.154
NPCD	-0.076	0.025	8.785	0.003
NROV	-0.062	0.019	10.070	0.001
NST	-0.577	0.394	2.146	0.142
intercept	8.203	24.730	0.110	0.740

Table 2. Model obtained by statistical analysis based on 11 metrics on INDUSTRIAL data set.

count the number of data structures.

5.2 Discriminant Analysis results

Discriminant analysis on risk-threshold method is based on two phases: training and validation. The risk-level of the metrics (MRL) is estimated to create the model during the training phase. This activity is performed by using a training data set different from the one used for validation. In the training phase, the risk-threshold was also identified on the basis of the processed files and it affected the classification of files in the validation data set.

On both data sets, using the whole metric sets contribution, results are not so good as expected: the models classified about 90% of modules in each group, but the correctness was around 70%. In this case, the high number of metrics failed in producing a suitable model for fault proneness estimation/prediction, due to the low risk level of some metrics; the classification correctness which has been achieved from INDUSTRIAL is 68%, where only one half of the files with faults have been identified.

Statistical coeff.	result	theoretic
Bravais-Pearson	0.662	1
Sensibility	0.740	1
Correctness	0.906	1
Completeness	0.831	1
Classification Corr.	0.842	1
Type I Errors	0.057	0
Type II Errors	0.100	0

Metric	β	Std. Error	Wald	p-value
NSLT	0.031	0.005	40.914	0.000
NGCL	-0.026	0.005	32.280	0.000
NDR	-0.006	0.002	12.744	0.000
CHUNKS	-0.140	0.048	8.519	0.004
DATADECLS	-0.089	0.041	4.866	0.027
intercept	-3.722	0.504	54.629	0.000

Table 3. Model obtained by statistical analysis based on 5 metrics on MEDICAL data set.

On the basis of the obtained results, all the available metrics have been grouped in 3 clusters on the basis of their main nature:

- Cohesion and complexity (18 metrics),
- Size (34 metrics),
- Volume and structure (36 metrics).

The Discriminant Analysis of risk-threshold on the three clusters produced significantly different results. The first cluster has been identified to be less satisfactory in predicting fault with respect to the others. The second and the third clusters produced similar results. On the other hand, still using a large set of metrics, they have not produced acceptable rates comparable to those obtained by using the logistic regression.

For the above reasons, as a first step, cluster 1 has been excluded from the analysis. On the basis of the results obtained by the training phase, some metrics contributed very marginally to the risk level and thus they were excluded. The remaining metrics were those more related to the highest MRLs. The results of this reduction have been two clusters of 14 metrics. In this case, the obtained models have been capable to estimate the fault-proneness with a rate of about the 70%. The details concerning the validation results for the INDUSTRIAL and MEDICAL data sets for the two clusters are reported in Tab. 4.

INDUSTRIAL - $2^{nd}cluster$

	Actual		
Model	no fault	fault	% correct
no fault	77	20	79.38%
fault	11	13	54.17%
		total	74.38%

13/134 not classified (90.3% classification)

MEDICAL - $2^{nd}cluster$

	Actual		
Model	no fault	fault	% correct
no fault	70	27	72.16%
fault	11	17	60.70%
		total	69.60%

6/131 not classified (95.4% classification)

INDUSTRIAL - $3^{rd}cluster$

	Actual		
Model	no fault	fault	% correct
no fault	78	20	78.57%
fault	11	19	47.83%
		total	72.73%

13/134 not classified (90.3% classification)

MEDICAL - $3^{rd}cluster$

	Actual		
Model	no fault	fault	% correct
no fault	67	21	76.13%
fault	12	25	67.50%
		total	73.60%

6/131 not classified (95.4% classification)

Table 4. Results of the risk-threshold analysis applied on clusters 2 and 3 of 14 metrics each.

The same method for reducing the dimension of the metric set based on MRL has been applied to select the metrics which are more related to risk of faults. After this second phase of reduction, two clusters of 5 metrics have been identified. They have produced satisfactory validation results on both INDUSTRIAL and MEDICAL data sets as shown in Tab. 5.

Two different models based on different groups of metrics have been produced by using MEDICAL and INDUSTRIAL data sets, while these models are very similar for the presence of 4 identical metrics on 5. The results of the evaluation are reported in Tab. 6. Observing the reduced clusters of metrics (see Tab. 6 for MRLs) which built the models, it is evident that some of the identified metrics are in common for their dominance on risk contribution: more precisely, what was found out is that 4 metrics (NAIV, NDR, NST, BYTES) are shared in common in the models produced from the processing of MEDICAL and INDUSTRIAL data sets, respectively.

INDUSTRIAL

	Actual		
Model	no fault	fault	% correct
no fault	85	20	80.95%
fault	6	12	78.86%
		total	76.03%

11/134 not classified (91.79% classification)

MEDICAL

	Actual		
Model	no fault	fault	% correct
no fault	68	23	74.70%
fault	11	23	67.60%
		total	72.80%

6/131 not classified (95.4% classification)

Table 5. Results of the risk-threshold analysis method applied on reduced clusters 2+3 of 5 metrics each.

5.3 Comparing Statistical and Discriminant Analyses

The obtained models for fault-proneness estimation have produced results which are in the range of those already obtained in past experiments performed by other research groups, when small set of metrics are used [7], [11], [15], etc. On the other hand, the models have highlighted some common metrics and their corresponding behaviours.
The models produced by Statistical Analysis (logistic regression) have the following properties:

- estimate fault proneness with a rate of about 80% on both data sets;
- based on 11 metrics for INDUSTRIAL sources and 5 metrics for MEDICAL data sets;
- include 5 metrics which are in common between the two models identified starting from the two

Statistical coeff.	result	theoretic
Bravais-Pearson	0.390	1
Sensibility	0.500	1
Correctness	0.860	1
Completeness	0.676	1
Classification Corr.	0.752	1
Type I Errors	0.090	0
Type II Errors	0.190	0

	MRL	
Metric	MEDICAL	INDUSTRIAL
NDR	0.353	0.860
NAIV	0.350	0.865
BYTES	0.344	0.714
NST	0.370	0.845
NII	3.548	-
SPNLIVEVAR	-	0.785

Table 6. Models obtained by discriminant analysis on risk-threshold of 5 metrics each on MEDICAL and INDUSTRIAL data sets.

data sets.

The models produced by Discriminant Analysis (risk-threshold) have the following properties:

- predict fault proneness on the validation subsets with a rate of about 78% on INDUSTRIAL data set and 72% on MEDICAL data set;
- based on 5 metrics for both data sets;
- include 4 metrics which are in common between the two models identified starting form the two data sets.

This analysis suggested to consider the metrics, which have been identified by the two models, as the most relevant for fault-proneness estimation and prediction. In this joined group of metrics, two of them (NDR, NAIV) are present in both. Thus, the resulted set was a group of 7 metrics that was supposed to be suitable to build a model for fault-proneness estimation/prediction, with less dependence on the software project.

6 Conclusions

This paper reported a research study begun with the analysis of about 100 metrics with the aim of identifying suitable models for the fault-proneness estimation and prediction of software modules/files. The objective has been to find out a compromise between the fault-proneness estimation rate and the size of the model in terms of number of metrics used and to compare two different methodologies for the generation of fault-proneness models. To this end, different methodologies have been used and compared, and their synergies exploited. The metodologies which have been taken into account are the statistic analysis (PCA and different forms of logistic regression) and the discriminant analysis (as risk-threshold analysis). They have been used to create models for fault-proneness estimation and prediction by using metrics addressing different aspects of computer programming: complexity, size, quality, cohesion, etc. These models have been cross validated by using data coming from two different projects of different companies that work in the telecommunication and medical areas.

These produced models have been cross validated by using data sets produced from source codes provided by two firms. The models include metrics related to software complexity, data size, general size, and cohesion among modules. This seems to be very interesting since the model has a quite good prediction rate compared to most of the models proposed in the literature and the rational could be the adoption of a model that takes into account a larger number of factors including the data size. The comparison with other similar models presented in the literature confirms the relevance of the proposed models in terms of accuracy of the results.

Acknowledgments

The authors thank the two companies that provided the source codes mentioned in the article as INDUSTRIAL and MEDICAL. A special thanks to Dr. F. Cinotti and Dr. L. Bruciamacchie, for their help in producing the huge amount of experiments and processing the code. A sincere thanks to Harry Sneed who provided the CPP-Analyzer tool for these experiments. Finally the authors thank Prof. M. Pighin for the valuable details and suggestions regarding risk-threshold method.

References

[1] V. R. Basili, L. Briand, and W. L. Melo. A valida-

tion of object oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, pp.751–761, October 1996.

[2] L. C. Briand, V. R. Basili, and C. J. Hetmanski. Developing interpretable models with optimized set reduction for identifying high-risk software components. IEEE Transactions Software Engineering, vol.19, n.11, pp. 1028-1044, November 1993.

[3] L. C. Briand, J. Wust, J. W. Daly, and D. V. Porter. Exploring the relationships between design measures and software quality in object oriented systems. Journal of Systems and Software, 1998.

[4] G. Dunteman, Principal Component Analysis, Sage University Paper, 07-69, Thousand Oaks, CA, USA, 1989.

[5] C. Ebert. Classification techniques for metric-based software development. Software Quality J., vol.5, no.4, pp.255-272, December 1996.

[6] F. Fioravanti, P. Nesi, and S. Perlini. Assessment of system evolution through characterization. In Proc. of the IEEE International Conference on Software Engineering, pp.456–459, Kyoto, Japan, April 1998.

[7] F. Fioravanti, P. Nesi. A Study on Fault-Proneness Detection of Object-Oriented Systems, In Proc. of the Fifth European Conference on Software Maintenance and Reengineering, IEEE Press, pp.121–130, Lisbon, Portugal, March 2001.

[8] H. F. Kaiser. The varimax criterion for analytic rotation in factor analysis. Psychometrika, vol.32, pp.187–198, 1958.

[9] R. Hochman, T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, S. J. Aud. Application of neural networks to software quality modeling of a very large telecommunications system. Neural Networks, IEEE Transactions on,vol.8, n.4,pp.902–909, July 1997 .

[10] T. M. Khoshgoftaar and E. B. Allen. Logistic regression modeling of software quality. Int. J. Reliability, Quality, and Safety Engineering., vol.6, n.4, pp.303-317, December 1999.

[11] T. M. Khoshgoftaar, E. B. Allen, D. Jianyu. Using regression trees to classify fault-prone software modules. IEEE Transactions on Reliability,vol.51 ,n.4, pp.455-462,December 2002.

[12] J.C. Munson, T.M. Khoshgoftaar. The Detection of Fault-prone Programs. IEEE Transactions on Software Engineering, vol.18, n.5, pp.423–432, 1992.

[13] N. Ohlsson, M. Zhao, and M. Helander. Application of multivariate analysis for software fault prediction. Software Quality Journal, vol. 7, pp.51-66, 1998.

[14] D. B. Simmons (editor), N. C. Ellis, H. Fujihara, W. Kuo. Software Measurement: A Visualization Toolkit For Project Control and Process Improvement Prentice Hall. November, 1997. ISBN 0-13-840695-2.

[15] M. Pighin, R. Zamolo. A predictive metric based on discriminant statistical analysis. In Proc. of the 19th International Conference on Software Engineering, ICSE 97. Boston, Massachusetts, USA, pp.262–270, May 1997.

[16] N. F. Schneidewind. Software metrics model for integrating quality control and prediction. in Proc. 8th Int. Symp. Software Reliability Engineering, pp.402-415, 1997.

[17] H. M. Sneed. Estimating the costs of software maintenance tool. In Proc. International Conference on Software Maintenance, pp.168-181, Opio, France, October 1995.