Attribute Selection in Software Engineering Datasets for Detecting Fault Modules

D. Rodríguez[1], R. Ruiz[2], J. Cuadrado-Gallego[1], J. Aguilar-Ruiz[2], M. Garre[1]

[1]Dept of Computer Science
The University of Alcalá
Ctra. Barcelona km 37,1
28805 Alcalá de Henares, Madrid, Spain

[2]Dept of Computer Science
University Pablo de Olavide
Ctra. Utrera km. 1
41013 Seville, Spain

{daniel.rodriguezg@,jjcg,miguel.garre}uah.es
{robertoruiz,aguilar}@upo.es

Abstract

Decision making has been traditionally based on managers experience. At present, there is a number of software engineering (SE) repositories, and furthermore, automated data collection tools allow managers to collect large amounts of information, not without associated problems. On the one hand, such a large amount of information can overload project managers. On the other hand, problems found in generic project databases, where the data is collected from different organizations, is the large disparity of its instances. In this paper, we characterize several software engineering databases selecting attributes with the final aim that project managers can have a better global vision of the data they manage. In this paper, we make use of different data mining algorithms to select attributes from the different datasets publicly available (PROMISE repository), and then, use different classifiers to defect faulty modules. The results show that in general, the smaller datasets maintain the prediction capability with a lower number of attributes than the original datasets.

1 Introduction

Successful software engineering (SE) projects need different types of estimates using past project data. In the last decade, several organizations have started to collect project management data so that companies without historical datasets can use these generic databases for estimation. For example, the ISBSG[1] collects project management data from multiple organizations; its latest release has data on over 4,000 projects and the information for each project is composed of more than 60 attributes. covering all phases is in the Software Development Life Cycle (SDLC). Furthermore, current Integrated Development Environments (IDE) and other metrics tools allow developers to gather large amounts of data from software modules and components. Examples of different types of data collected has been made available through the PROMISE repository[2].

With such large data repositories, data mining techniques and tools designed to support the extraction in an automatic way the information useful for decision making or exploration of the data source. Since data may not be organized in a way that facilitates the extraction of useful information, typical data mining processes are composed of the following steps:

- Data preparation. The data is formatted in a way that tools can manipulate it, merged from different databases, etc.
- Data mining. It is in this step when the automated extraction of knowledge from the data is carried out. Data mining techniques include classification as explained later.
- Proper interpretation of the results, including the use of visualization techniques.

Many software engineering problems like cost estimation and forecasting can be viewed as classification problems. A classifier resembles a function in the sense that it attaches a value (or a range or a description), named the *class*, C, to a set of attribute values A_1, A_2,... A_n, i.e. a classification function will assign a class to a set of descriptions based on the characteristics of the instances for each

[1]http://www.isbsg.org/

[2]http://promise.site.uottawa.ca/SERepository/

33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007)
0-7695-2977-1/07 $25.00 © 2007 IEEE

A_1:Size	...	A_n:Complexity	$Class$:No. of errors
a_{11}	...	a_{11}	c_1
...	...	...	...
a_{11}	...	a_{nm}	c_n

attribute, For example, as shown in Table 1, given the attributes size, complexity, etc., a classifier can be used to predict the number of errors of a module.

With such large datasets, part of the preprocessing consists of reducing the redundant and irrelevant attributes. This paper is focused on the application of different attribute selection techniques, also known as feature selection (FS), for identifying the most relevant attributes from different datasets. Then, the different learning algorithms can be applied to obtain more accurate estimates. Attribute selection is part of the data preparation phase to generate a reduced dataset which can be useful in different aspects:

- A reduced volume of data facilitates the application of different data mining or searching techniques to be applied.
- Irrelevant and redundant attributes can generate less accurate and more complex models. Furthermore, data mining algorithms can be executed faster.
- It is possible avoid data collection for those irrelevant and redundant attributes in the future.

The paper is organized as follows. Section 1.1 explains the background behind Feature Selection, learning Classifiers using Feature Selection and common techniques used in data mining for evaluation. Section 2 describes the experimental results using several datasets from the PROMISE repository. Finally, Section 3 concludes the papers and outlines future work.

1.1 Attribute Selection

The problem of feature selection received a thorough treatment in pattern recognition and data mining. Most of the feature selection algorithms tackle the task as a search problem, where each state in the search specifies a distinct subset of the possible attributes [2]. The search procedure is combined with a criterion to evaluate the merit of each candidate subset of attributes. There are a multiple possible combinations between each procedure search and each attribute measure [12].

There are various ways in which feature selection algorithms can be grouped according to the attribute evaluation measure: depending on the type (filter or wrapper techniques) or on the way that features are evaluated (individual or subset evaluation). The filter model relies on general characteristics of the data to evaluate and select gene subsets without involving any mining algorithm. The wrapper model requires one predetermined mining algorithm and uses its performance as the evaluation criterion. It searches for features better suited to the mining algorithm, aiming to improve mining performance. It is, however, more computationally expensive [11, 10] than filter models. Feature ranking (FR), also called feature weighting [2], assesses individual features and assigns them weights according to their degrees of relevance, while the feature subset selection (FSS) evaluates the goodness of each found feature subset.

1.2 Attribute Selection Process

Attribute selection algorithms designed with different evaluation criteria broadly fall into two categories [11]: the filter model and the wrapper model.

- The filter model relies on general characteristics of the data to evaluate and select feature subsets without involving any mining algorithm.
- The wrapper model requires one predetermined mining algorithm and uses its performance as the evaluation criterion. It searches for features better suited to the mining algorithm aiming to improve mining performance, but it also tends to be more computationally expensive than filter model [10, 11].

Another possibility, it is to arrange attributes according to a rank. In the feature ranking algorithms category, one can expect a ranked list of features which are ordered according to evaluation measures. A subset of features is often selected from the top of a ranking list. A feature is good and thus will be selected if its weight of relevance is greater than a user-specified threshold value, or we can simply select the first k features from the ranked list. This approach is efficient for high-dimensional data due to its linear time complexity in terms of dimensionality.

Some existing evaluation measures that have been shown effective in removing both irrelevant and redundant features include the consistency measure [4], the correlation measure [6] and the estimated accuracy of a learning algorithm [10].

- Consistency measure attempts to find a minimum number of features that separate classes as consistently as the full set of features can. An inconsistency is defined as to instances having the same feature values but different class labels.

- Correlation measure evaluates the goodness of feature subsets based on the hypothesis that good feature subsets contain features highly correlated to the class, yet uncorrelated to each other.

- Wrapper-based attribute selection uses the target learning algorithm to estimate the worth of attribute subsets. The feature subset selection algorithm conducts a search for a good subset using the induction algorithm itself as part of the evaluation function.

Langley [11] notes that feature selection algorithms that search through the space of feature subsets must address four main issues: the starting point of the search, the organization of the search, the evaluation of features subsets and the criterion used to terminate the search. Different algorithms address theses issues differently.

It is impractical to look at all possible feature subsets, even if the size is small. Feature selection algorithms usually proceed greedily and be classified as follows:

- *Forward selection* techniques add features to an initially empty set.

- *Backward elimination* remove features from an initially complete set.

- *Hybrid* techniques both add and remove features as the algorithm progresses.

Forward selection is much faster than backward elimination and therefore scales better to large data sets. A wide range of search strategies can be used: best–first, branch–and–bound, simulated annealing, genetic algorithms (see Kohavi and John [10] for a review). In [4], different search strategies, namely exhaustive, heuristic and random search, are combined with consistency measure to form different algorithms. The time complexity is exponential in terms of data dimensionality for exhaustive search and quadratic for heuristic search. The complexity can be linear to the number of iterations in a random search, but experiments show that in order to find best feature subset, the number of iterations required is mostly at least quadratic to the number of features [4]. In [6], correlation measure is applied in an algorithm called CFS that exploit heuristic search (best first) to search for candidate feature subsets.

One of the most frequently used search techniques is hill-climbing (greedy). It starts with an empty set and evaluates each attribute individually to find the best single attribute. It then tries each of the remaining attributes in conjunction with the best to find the most suited pair of attributes. In the next iteration, each of the remaining attributes are tried in conjunction with the best pair to find the most suited group of three attributes. This process continues until no single attribute addition improves the evaluation of the subset; i.e., subset evaluator is run *M* times to choose the best single attribute, *M-1* times to find the best pair of attributes, *M-2* times the best group of three, and so on. For example, if we have chosen five attributes through this method, the subset evaluator has been run *M+(M-1)+(M-2)+(M-3)+(M-4)* times.

1.3 Learning Classifiers using Attribute Selection

In order to compare the effectiveness of attribute selection, feature sets chosen by each technique are tested with two different and well-known types of classifiers: an instance–base classifier (IB1) and a decision tree classifier (C4.5). These three algorithms have been chosen because they represent three quite different approaches to learning, and their long standing tradition in classification studies. Other techniques, such as neural nets, are not included among our classification models due to their low human-transparency.

- The IB1 [1] is a nearest-neighbor (k-NN) classifier. To classify a new test sample, all training instances are stored and the nearest training (usually Euclidean distance metric) instance regarding the test instance is found: its class is retrieved to predict this as the class of the test instance.

- C4.5 [14]. The C4.5 algorithm summarize training data in the form of a decision tree. Along with systems that induce logical rules, decision tree algorithms have proved popular in practice. This is due in part to their robustness and execution speed, and to the fact that explicit concept descriptions are produced, which users can interpret. Nodes in the tree correspond to features, and branches to their associated values. The leaves of the tree correspond to classes. To classify a new instance, one simply examines the features tested at the nodes of the tree and follows the branches corresponding to their observed values in the instance. Upon reaching a leaf, the process terminates, and the class at the leaf is assigned to the instance. C4.5 uses the gain ratio criterion to select the attribute to be at every node of the tree.

1.4 Evaluation

When evaluating the prediction accuracy of the classification methods we described above, it is important not to use the same instances for training and evaluation. Feature selection methods will perform well on examples they have seen during training. To get a realistic estimate of performance of the classifier, we must test it on examples that did not appear in the training set.

A common method to test accuracy in such situations is cross-validation. To apply this method, we partition the data into *k* sets of samples, $C_1, \ldots, C_k$ (typically, these will be of roughly the same size). Then, we construct a data set $D_i = D - C_i$, and test the accuracy of f_{D_i} on the samples in C_i. Having done this for all $1 \leq i \leq k$ we estimate the accuracy of the method by averaging the accuracy over the *k* cross-validation trials. Cross-validation has several important properties. First, the training set and the test set in each trial are disjoint. Second, the classifier is tested on each sample exactly once. Finally, the training set for each trial is $(k-1)/k$ of the original data set. Thus, for large *k*, we get a relatively unbiased estimate of the classifier behavior given a training set of size *m* [9]. There are several possible choices of *k*. A common approach is to set $k = m$. In this case, every trial removes a single sample and trains on the rest. This method is known as *leave one out cross validation* (LOOCV). Other common choices are $k = 5$ or $k = 10$. In this work we have used $k = 10$.

Another common way to measure the goodness of data mining applications is through the $f - measure$, which is defined by Eq. 1:

$$f - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \qquad (1)$$

where *precision* is defined as the proportion of the instances which truly have class x (true positives) among all those which were classified as class x(true and false positives; and *Recall* (also known as sensitivity or positive accuracy) is the proportion of the examples which truly have class x (true positives), i.e., true positives divided by the elements that belong to the class (true positives and false negatives).

2 Experimental Results

In this paper, we have applied feature selection to the CM1, JM1, KC1, KC2, and PC1 datasets available in the PROMISE repository [15], to generate models for defect classification. These datasets were created from projects carried out at NASA and collected under their metrics [3].

All datasets contain 22 attributes composed of 5 different lines of code measure, 3 McCabe metrics [13], 4 base Halstead measures [7], 8 derived Halstead measures [7], a branch-count, and the last attribute is 'problems' with 2 classes (false or true, wether the module has reported defects). For a comprehensive coverage and explanation of the metrics, we refer to Fenton and Pfleeger [5]. The number of instances, however, varies among the datasets. Table 2 shows the number of attributes for each of the datasets.

As stated previously, feature selection can be grouped into filter or wrapper depending on whether the classifier is used to find the subset. In this paper, for the filter model, we have used consistency and correlation measures; for the wrapper-method, two standard classifiers have been applied, the IB1 and C4.5 classifiers.

[3] http://mdp.ivv.nasa.gov/

Dataset	*No. of attributes*
CM1	498
JM1	10885
KC1	2109
KC2	522
PC1	1109

			Wrapper	
	CFS	CNS	C4.5	IB1
CM1	5.24	1.30	1.02	1.83
JM1	8.01	19.99	3.29	2.91
KC1	7.77	17.61	2.97	1.94
KC2	5.52	13.27	1.78	2.16
PC1	4.63	10.67	1.67	1.58

The experiments were conducted using algorithms implemented in the WEKA [16] toolkit, either using the Explorer or the Experimenter tools.

The results reported in this section were obtained with 10 runs, each run is a 10-fold cross-validation, i.e., in one run, an attribute subset was selected using the 90% of the instances, then, the accuracy of this subset was estimated over the unseen 10% of the data. This was performed 10 times, each time proposing a possible different feature subset. In this way, estimated accuracies, selected attribute numbers and time needed were the result of a mean over 10 times 10-cross-validation samples.

Table 3 shows the average number of attributes selected using correlation, consistency or the wrapper method. From the result, we can conclude that the wrapper method selects smaller sets of attributes on average but it is the more expensive computationally.

Table 4 shows the average percentage of correctly classified instances. The statistical *t-test* provided by the WEKA Experimenter tool [16] was used to investigate whether the estimation with the subset was statistically significant when compared with the original set of attributes (orig).

From the results, we can conclude that in general attribute selection improves the accuracy of the estimation, being the wrapper method superior to the filter method. The C4.5 classifier using the wrapper method obtained 3 statistically significant improved models.

The wrapper is a good option as either improves the ac-

curacy or when the accuracy is not improved is because the models generated are very simple (the number of selected attributes is very low). For example, for the JM1 dataset, the accuracy using IB1 wrapper is below the original accuracy (72.01% compared with the 79.73% when using all attributes); however, it just needs less than 3 attributes to obtain an acceptable accuracy.

In this case, with ten-cross validation, Table 4 shows the number of times that an attribute has been selected out of the 10 times that the algorithm was run using the CFS (correlation) with sequential search. We selected this algorithm based on correlations because it obtained good result with all classifiers. It possible to analyse this table in 2 dimensions. First, analysing columns, i.e, attributes selected for each dataset, and second, analyzing rows, the number of times that an attribute is selected across different datasets. For example, with JM1, 7 attributes were always selected (`loc`, `ev(g)`, `iv(g)`, `i`, `IOComment`, `IOBlank` and `IOCodeAndComment`), 4 times the algorithm selected `v(g)` and 6 times `branchCount`. On the row dimension, it can be concluded that the attribute `i` is is relevant and provides important information for classification, attribute `i` has been always selected in 4 datasets (CM1, JM1, KC1 and KC2) and 5 times in the PC1 dataset. In the KC2 dataset, the attribute *loc* has been selected 4 times, and the attribute *v(g)* was never selected.

3 Conclusions and Future Work

In general, the application of data mining techniques to software engineering (SE) is in its inception, and in particular, attribute selection has not yet been covered in in the SE domain by many researchers. Among these works, Chen et al [3] have analyzed the application of feature selection using wrappers to the problem of cost estimation. They also concluded that the reduced dataset could improve the estimation. Kirsopp and Shepperd [8] have also analyzed the application of attribute selection to cost estimation reaching similar conclusions.

In this paper, attribute selection techniques was applied to different datasets from the PROMISE repository before to improve the accuracy of different types of data mining classifiers to defect faulty modules. The results show that smaller datasets maintain the prediction capability and has a lower number of attributes than the original, yet it allow us to produce better or equal estimates. This is beneficial for project managers or quality engineers as a way of characterizing the dataset and better understanding of the data at hand.

We will extend this work with other data mining techniques and further SE datasets with their particular problems. For example, in this work, many datasets are unbalanced, i.e., many more modules were classified as non-defective than defective. Current research works on how to balance datasets can be applied before the techniques described in this paper.

Acknowledgements

We thank the Spanish Ministry of Education and Science for supporting this research (TIN2004-06689-C03).

References

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[3] Z. Chen, T. Menzies, D. Port, and B. Boehm. Finding the right data for software cost modeling. *IEEE Software*, 22:38–46, 2005.

[4] M. Dash, H. Liu, and H. Motoda. Consistency based feature selection. In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 98–109, 2000.

[5] N. E. Fenton and S. L. Pfleeger. *Software metrics: a Rigorous & Practical Approach*. International Thompson Press, 1997.

[6] M. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, Department of Computer Science, Hamilton, New Zealand, 1999.

[7] M. H. Halstead. *Elements of software science*. Elsevier Computer Science Library. Operating And Programming Systems Series; 2. Elsevier, New York ; Oxford, 1977.

[8] C. Kirsopp and M. Shepperd. Case and feature subset selection in case-based software project effort prediction.

[9] R. Kohavi and G. John. Automatic parameter selection by minimizing estimated error. In *12th Int. Conf. on Machine Learning*, pages 304–312, San Francisco, 1995.

[10] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 1-2:273–324, 1997.

[11] P. Langley. Selection of relevant features in machine learning. In *Procs. Of the AAAI Fall Symposium on Relevance*, pages 140–144, 1994.

[12] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Eng.*, 17(3):1–12, 2005.

[13] T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, December 1976.

[14] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, California, 1993.

[15] J. S. Shirabad and T. J. Menzies. The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[16] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.

	C4.5				IB1			
Dataset	Orig.	CFS	CNS	WRP	Orig.	CFS	CNS	WRP
CM1	88.05	89.30	89.82	90.16○	84.98	83.59	83.86	86.44
JM1	79.73	80.83○	79.72	80.78○	75.91	75.54	75.91	72.01
KC1	84.04	84.54	84.22	84.80	83.35	82.77	83.59	73.80
KC2	81.19	83.64	82.18	84.44○	78.62	77.64	78.10	76.45
PC1	93.63	93.17	93.10	92.87	91.87	91.42	91.18	85.96

○, statistically significant improvement

	C4.5				IB1			
Dataset	Orig.	CFS	CNS	WRP	Orig.	CFS	CNS	WRP
CM1	0.94	0.94	0.95	0.95○	0.92	0.91	0.88	0.92
JM1	0.88	0.89○	0.88	0.89○	0.85	0.85	0.85	0.79
KC1	0.91	0.91	0.91	0.92	0.90	0.90	0.90	0.78
KC2	0.88	0.90	0.89	0.91○	0.86	0.86	0.86	0.84
PC1	0.97	0.96	0.96	0.96	0.96	0.95	0.95	0.89

○, statistically significant improvement

	CFS-sf				
	CM1	JM1	KC1	KC2	PC1
loc: McCabe's line count of code	6	10	1	4	1
v(g): McCabe "cyclomatic complexity"	0	4	4	0	2
ev(g): McCabe "essential complexity"	0	10	2	10	0
iv(g): McCabe "design complexity"	7	10	1	0	0
n: Halstead total operators + operands	0	0	1	2	0
v: Halstead "volume"	0	0	0	0	0
l: Halstead "program length"	0	0	0	0	0
d: Halstead "difficulty"	1	0	8	2	0
i: Halstead "intelligence"	10	10	10	10	5
e: Halstead "effort"	0	0	4	0	0
b: Halstead	1	0	0	0	0
t: Halstead's time estimator	0	0	2	0	0
IOCode: Halstead's line count	1	0	6	1	1
IOComment: Halstead's lines of comments	10	10	9	2	9
IOBlank: Halstead's blank lines	4	10	10	2	10
IOCodeAndComment	0	10	0	4	10
uniq-Op: unique operators	6	0	1	8	0
uniq-Opnd: unique operands	4	0	7	10	4
total-Op: total operators	0	0	0	0	1
total-Opnd: total operands	0	0	4	0	0
branchCount: of the flow graph	0	6	8	0	0