November 2009

High-Dimensional Software Engineering Data and Feature Selection

Huanjing Wang
Western Kentucky University, huanjing.wang@wku.edu

Taghi M. Khoshgoftaar
Florida Atlantic University, taghi@cse.fau.edu

kehan Gao
Eastern Connecticut State University, kgaok@easternct.edu

2009 21st IEEE International Conference on Tools with Artificial Intelligence

High-Dimensional Software Engineering Data and Feature Selection

Huanjing Wang
Western Kentucky University
huanjing.wang@wku.edu

Taghi M. Khoshgoftaar
Florida Atlantic University
taghi@cse.fau.edu

Kehan Gao
Eastern Connecticut State University
gaok@easternct.edu

Naeem Seliya
University of Michigan – Dearborn
nseliya@umich.edu

Abstract

Software metrics collected during project development play a critical role in software quality assurance. A software practitioner is very keen on learning which software metrics to focus on for software quality prediction. While a concise set of software metrics is often desired, a typical project collects a very large number of metrics. Minimal attention has been devoted to finding the minimum set of software metrics that have the same predictive capability as a larger set of metrics – we strive to answer that question in this paper. We present a comprehensive comparison between seven commonly-used filter-based feature ranking techniques (FRT) and our proposed hybrid feature selection (HFS) technique. Our case study consists of a very high-dimensional (42 software attributes) software measurement data set obtained from a large telecommunications system. The empirical analysis indicates that HFS performs better than FRT; however, the Kolmogorov-Smirnov feature ranking technique demonstrates competitive performance. For the telecommunications system, it is found that only 10% of the software attributes are sufficient for effective software quality prediction.

Keywords: *software metrics, quality prediction, feature ranking, hybrid feature selection, high-dimensional data.*

1 Introduction

In software quality prediction, the quality and characteristics of the underlying software measurement data plays an important role in the efficacy of the prediction model. One aspect of such characteristic is "which software metrics are good predictors for a given software system?". A typical mid- to large-scale software project collects several software metrics as candidates for defect prediction [6]. However, it is likely that many of them provide redundant information, or provide no information, or in some cases, have an adverse effect on the prediction model. This study strives to answer the practical question of "for a given software project, what is minimum number of software metrics that should be considered for building a defect prediction model?"

A typical software quality prediction model is trained using software metrics (independent variables) and fault data (dependent variable) that have been collected from previously-developed software releases or similar projects. Subsequent to model evaluation, the quality of currently-under-development program modules can be estimated: for example, fault-prone (*fp*) or not-fault-prone (*nfp*). Such a software quality model has been the subject of intensive research [19, 22, 27]. However, very little attention has been given to the problem of attribute selection in software measurement data for defect prediction. Some studies have shown that the performance of software quality prediction model can be improved when irrelevant and redundant features are eliminated from the original software measurement data set [9, 15, 24].

We explore the data mining concept of feature selection and investigate those technologies in the context of software quality prediction and software metrics. Various techniques developed from data mining and machine learning have been successfully applied for deriving new information in a variety of domains [4, 25]. Feature selection (or attribute selection) has become a vital pre-processing step in most data mining and machine learning problems. In addition to improving the quality of the machine learning data set, feature selection is particularly useful for high-dimensional data – for example, software measurement data sets considered in this study. The aim of attribute selection is to find a feature subset (i.e., data reduction) that can learn and describe the data set such that it is equivalent to the same task being done by the original data set (i.e., without any data reduction).

1082-3409/09 $26.00 © 2009 IEEE
DOI 10.1109/ICTAI.2009.20

The two general categories for feature selection are *filters* and *wrappers*. Filters are algorithms in which a feature subset is selected without involving any learning (classifier) algorithm. Wrappers are algorithms that use feedback from a learning algorithm to determine which feature(s) to include in building a classification model. Another categorization for feature selection techniques is *feature ranking techniques* and *feature subset selection techniques*. Feature ranking ranks the attributes according to their individual predictive power, while feature subset selection approaches select subsets of attributes that collectively have good predictive power. From a software engineering point of view, feature selection can reduce the time for metrics collection, model calibration, and model evaluation of future software development efforts of similar systems.

The focus of this paper is to evaluate several feature selection techniques, including seven filter-based ranking techniques (FRT) and our proposed hybrid feature selection (HFS) method. The HFS method consists of a feature ranking technique followed by a consistency-based feature subset selection, i.e., automatic hybrid search (AHS) [16, 20]. The seven feature ranking techniques considered are: chi-square (CS), information gain (IG), gain ratio (GR), Kolmogorov-Smirnov statistic (KS), two forms of the ReliefF algorithm (RLF), and symmetrical uncertainty (SU). Our empirical study of the different feature selection techniques will answer the question posted earlier, i.e. "what is minimum number of software metrics that should be considered for building a defect prediction model for a given software project?" The aim is to do so without degrading the generalization power of the software quality prediction model.

The answer to the above question is particularly important because of the high-dimensionality of the software measurement data of our case study. The four consecutive releases of a very large telecommunications system are considered as case study data, and include 42 software metrics and defect data collected for every program module. The software quality prediction models are built using five different classification algorithms [28]: naive Bayes, multilayer perceptrons, K-nearest-neighbor, support vector machine, and logistic regression. One of the project releases is considered as training data, while the other three releases are considered as test data.

The remainder of the paper is organized as follows. We review relevant literature on feature selection in Section 2. Section 3 provides detailed information about the filter-based feature ranking techniques, the proposed hybrid feature selection algorithm, the five classifiers, and the classifier performance metric used in our study. Section 4 provides a description of the case study data sets, and Section 5 presents empirical results of our study. Finally, we conclude the paper in Section 6, and provide suggestions for future work.

2 Related Work

This section provides a brief coverage on key feature selection works in the fields of data mining and software engineering. An exhaustive coverage is avoided due to space considerations.

Liu and Yu [21], provide a survey of feature selection algorithms and present an integrated approach to intelligent feature selection. Guyon and Elisseeff [8] outline key approaches used for attribute selection, including feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods. Hall and Holmes [9] investigated six attribute selection techniques that produce ranked lists of attributes and applied them to several data sets from the UCI machine learning repository. Jong et al. [14] introduced methods for feature selection based on support vector machines. Ilczuk et al. [12] highlighted the importance of attribute selection in judging the qualification of patients for cardiac pacemaker implantation. In the context of text mining, where attributes are binary in value, Forman [7] investigates multiple filter-based feature ranking techniques. Most of the above works have focuses on feature selection with categorical data or binary data; in contrast, this paper focuses on feature selection with numerical/continuous data.

Rodríguez et al. [24, 25] applied attribute selection with three filter models and three wrapper models to five software engineering data sets. It was stated that the wrapper model was better than the filter model; however, that came at a very high computational cost. Their conclusions were based on evaluating models using cross-validation instead of an independent test data set. It is known in the software engineering community that prediction models are best evaluated based on their generalization performance, i.e., using a test data set. In our study, three independent test data sets are used for evaluating the different prediction models.

3 Methodology

3.1 Filter-Based Feature Ranking techniques

Feature ranking assesses attributes individually and ranks attributes according to their individual predictive power. Filter feature ranking techniques (FRT) rank features independently without involving any learning algorithm that will use the selected features. The procedure of feature ranking is to score each feature according to a particular method, allowing the selection of the best set of features. The advantage of feature ranking is that it requires

only the computation and sorting of the scores of each feature individually. We discuss the feature ranking techniques investigated in our study: chi-square (CS), information gain (IG), gain ratio (GR), two types of ReliefF (RFF and RFT), symmetrical uncertainty (SU), and Kolmogorov-Smirnov method (KS). For specific algorithmic details on these techniques, the reader is referred to the various cited references.

The chi-square (CS) [3] test is used to examine if there is 'no association' between two attributes, i.e. whether the two variables are independent. CS is more likely to find significance to the extent that (1) the relationship is strong, (2) the sample size is large, and/or (3) the number of values of the two associated features is large. Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy, which is based on information theory [28]. For binary class problem, such as *fp* and *nfp*, the entropy is 0 if there is at most one class present and the entropy is 1 (at its maximum) when the proportions of all presented classes are equal.

Information gain (IG) [28] is the information provided about the target class attribute Y, given the value of other attribute X. Information gain measures the decrease of the weighted average impurity of the partitions, compared with the impurity of the complete set of data. A drawback of IG is that tends to prefer attributes with a larger number of possible values, i.e, if one attribute has a larger number of values, it will appear to gain more information than those with fewer values, even if they are actually no more informative. One strategy to counter this problem is to use the gain ratio (GR), which penalizes multiple-valued attributes. Symmetrical uncertainty (SU) [28] is another way to overcome the problem of IG's bias toward attributes with more values, and it does so by dividing it by the sum of the entropies of X and Y.

Relief is an instance-based feature ranking technique introduced by Kira and Rendell[17]. ReliefF is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the WEKA tool [28]. When the 'weightByDistance' (weight nearest neighbors by their distance) parameter was set as default (false), the algorithm is referred to as RFF; when the parameter was set to true, the algorithm is referred to as RFT. The Kolmogorov-Smirnov Method (KS) is a feature selection method recently proposed by our research team [16, 20]. It utilizes the Kolmogorov-Smirnov statistic to measure the maximum differences between the empirical distribution function of the posterior probabilities of instances in each class. The features can be ranked based on their KS scores, and then selected according to the number of features needed.

3.2 Proposed hybrid feature selection method

Feature subset selection techniques search the set of possible features as a group and evaluate their collective suitability. The Automatic Hybrid Search (AHS), our recently proposed feature subset selection search method [16, 20], uses the consistency rate (CR) properties. AHS relies on the monotonic property of consistency rate, which has the following facts: (1) the complete attribute set (D) has the highest consistency rate δ, i.e. the consistency rate of any attribute subset is less than or equal to δ; (2) the superset of a consistent attribute subset is also consistent; and (3) if $\mathrm{CR}(S_i, D) \leq \mathrm{CR}(S_j, D)$, then $\mathrm{CR}(S_i \cap f, D) \leq \mathrm{CR}(S_j \cap f, D)$, where f is an attribute not in S_i and S_j.

The AHS algorithm works as follows. The consistency rate of the complete attribute set is computed first, and then starting with size 1 of any attribute, the attribute subsets that have the locally highest consistency rate are selected. These selected attribute subsets will be used to generate supersets. The process is repeated until finding the attribute subsets that have the same consistency rate or the specified number of attributes is reached. If more than one attribute subsets are generated, the C4.5 [23] classifier will be used to decide which attribute subset is selected based on an error rate.

We proposed a new hybrid feature selection (HFS) method which is a combination of a filter-based feature ranking technique and a consistency-based feature subset selection algorithm, AHS. The proposed HFS method works as follows: the top 30% of the features are selected from the full feature set using filter-based feature ranking techniques. Thus, the original data set is reduced and this reduced data set is then the input to AHS. A subset of k features with highest local consistency rate is selected. In our study, we vary k with values of 2, 3, 4, and 6. These values may be different for another software project, since the subset size is likely to depend on the application domain and project characteristics. The overall structure of HFS is presented in Figure 1.

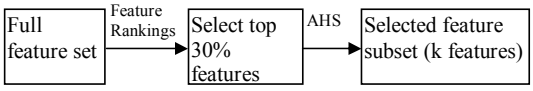


Figure 1. Hybrid Feature Subset Selection Method

3.3 Classifiers

Software quality prediction models are built with five different classification algorithms, including naive Bayes (NB) [13], multilayer perceptron (MLP) [10], K-nearest

neighbors (KNN) [1], support vector machine (SVM) [26], and logistic regression (LR) [18]. These were selected because of their common use in software engineering and data mining, and also because they do not have a built-in feature selection capability. Unless stated otherwise, we use default parameter settings for the different learners as specified in the WEKA data mining tool [28]. Parameter settings are changed only when a significant improvement in performance is obtained.

In the case of MLP, the 'hiddenLayers' parameter was set to '3' to define a neural network with one hidden layer containing three nodes, and the 'validationSetSize' parameter was set to '10' to cause the classifier to leave 10% of the training data aside as a validation set to determine when to stop the iterative training process. For the KNN classifier, the 'distanceWeighting' parameter was set to 'Weight by 1/distance', the 'kNN' parameter was set to '30', and the 'crossValidate' parameter was set to 'true'. In addition, the algorithms was modified slightly so that it chooses the k which produces the highest mean of the true positive rate and true negative rate. In the case of SVM, the 'complexity constant c' was set to '5.0' and 'build Logistic Models' was set to 'true'.

3.4 Performance Evaluation

A two-group classification problem, such as fault-prone and not-fault-prone, has four possible prediction outcomes: true positive (TP) (i.e., correctly classified positive instance), false positive (FP) (i.e., negative instance classified as positive), true negative (TN) (i.e., correctly classified negative instance), and false negative (FN) (i.e., positive instance classified as negative). The four values form the basis for several other performance measures that are well known and commonly used for classifier evaluation.

The performance measure used in our study is the Area Under the ROC (Receiver Operating Characteristic), abbreviated as AUC. The AUC is a single-value measurement, whose value ranges from 0 to 1. The ROC curve is used to characterize the trade-off between hit (true positive) rate and false alarm (false positive) rate [5]. The true positive rate is computed as $\frac{|TP|}{|TP|+|FN|}$, while the false positive rate is computed as $\frac{|FP|}{|FP|+|TN|}$. A classifier that provides a large area under the curve is preferable over a classifier with a smaller area under the curve.

4 Software Measurement Data Description

The software metrics and defect data for this case study (denoted as LLTS) was collected from a very large legacy telecommunications software system. The software, comprising of several million lines of code, was developed in a large organization by professional programmers using a proprietary high level procedural language, PROTEL [11]. A decision support system for software measurements and software quality modeling was periodically used to measure the static attributes of the most recent version of the code. The software measurement data sets used in this study consists of 42 software metrics [11], including 24 product metrics, 14 process metrics, and 4 execution metrics – these metrics are not shown due to paper size considerations. The dependent variable is the class of the program module, *fp* (fault-prone) or *nfp* (not fault-prone). A module with one or more faults is considered *fp*, and *nfp* otherwise.

The software measurement data sets consists of four successive releases of the LLTS system. The four data sets are labeled as SP1, SP2, SP3 and SP4, where each release is characterized by the same number and type of software attributes, but has a different number of instances (program modules). The SP1, SP2, SP3 and SP4 data sets consisted of 3649, 3981, 3541, and 3978 program modules, respectively. A unique characteristic of these data sets is that they all suffer from class imbalance, where the proportion of *fp* modules is much lower that the *nfp* modules. The proportions of *nfp* modules of SP1, SP2, SP3 and SP4 are 93.72%, 95.25%, 98.67%, and 97.69%, respectively. The SP1 data set is used for training purposes, while the SP2, SP3, and SP4 data sets are used for testing the different software quality prediction models.

Table 1. FRT vs. HFS – NB

		Size 2		Size 3		Size 4		Size 6	
		FRT	HFS	FRT	HFS	FRT	HFS	FRT	HFS
SP2	CS	0.6612	0.7223	0.7324	0.7209	0.7557	0.7312	0.7850	0.7409
	GR	0.6612	0.7324	0.6992	0.7724	0.7484	0.7701	0.8275	0.8181
	IG	0.8198	0.7834	0.8199	0.7733	0.8104	0.8186	0.8119	0.8158
	RFF	0.7839	0.7324	0.7997	0.7696	0.7965	0.7966	0.7955	0.8174
	RFT	0.7839	0.7528	0.7797	0.7607	0.7965	0.7828	0.7929	0.8166
	SU	0.6815	0.7324	0.8123	0.7724	0.8199	0.8259	0.8261	0.8309
	KS	0.8109	0.7554	0.8116	0.8218	0.8114	0.8219	0.8125	0.8230
SP3	CS	0.6420	0.7170	0.7449	0.7300	0.7502	0.7604	0.7330	0.7523
	GR	0.6420	0.7385	0.7072	0.7633	0.7519	0.7643	0.8193	0.8079
	IG	0.8380	0.7695	0.8330	0.8072	0.8388	0.8442	0.8242	0.8284
	RFF	0.8439	0.7385	0.8419	0.7680	0.8380	0.7759	0.8440	0.8258
	RFT	0.8439	0.7584	0.8399	0.7773	0.8380	0.7602	0.8317	0.8203
	SU	0.7105	0.7385	0.8280	0.7633	0.8308	0.8158	0.8089	0.8106
	KS	0.8311	0.7569	0.8227	0.8282	0.8313	0.8244	0.8225	0.8151
SP4	CS	0.6503	0.6654	0.7301	0.6861	0.7449	0.7091	0.7697	0.7257
	GR	0.6503	0.7702	0.6801	0.7939	0.7258	0.7875	0.8113	0.8257
	IG	0.8216	0.7853	0.8220	0.7848	0.8315	0.8318	0.8208	0.8225
	RFF	0.7799	0.7702	0.7966	0.7696	0.7933	0.8043	0.8058	0.8303
	RFT	0.7799	0.7593	0.7711	0.7371	0.7933	0.7748	0.7988	0.8147
	SU	0.6925	0.7702	0.7950	0.7939	0.8269	0.8297	0.8266	0.8226
	KS	0.7908	0.7832	0.7892	0.8231	0.8236	0.8192	0.8131	0.8124

5 Empirical Results

In order to evaluate the appropriateness of the k selected attributes (the minimum set of software metrics for defect prediction), we varied k with values of 2, 3, 4, and 6. For the LLTS system, we chose $\lceil \log_2 42 \rceil$ as the upper limit of range for varying the value of k, after consulting a software engineering domain expert with more than 20 years experience in the area of software quality engineering. The

Table 2. FRT vs. HFS – MLP

		Size 2		Size 3		Size 4		Size 6	
		FRT	HFS	FRT	HFS	FRT	HFS	FRT	HFS
SP2	CS	0.6612	0.7456	0.7384	0.7332	0.7600	0.6945	0.7886	0.6939
	GR	0.6612	0.7701	0.6091	0.8031	0.7930	0.8095	0.8190	0.8308
	IG	0.8241	0.8130	0.8201	0.8205	0.8244	0.8330	0.8185	0.8310
	RFF	0.8163	0.7701	0.8006	0.7744	0.8054	0.7972	0.8116	0.8217
	RFT	0.8163	0.7633	0.8006	0.7758	0.8054	0.7901	0.8072	0.8258
	SU	0.8122	0.7701	0.8149	0.8031	0.8234	0.8256	0.8277	0.8378
	KS	0.8142	0.7963	0.8120	0.8222	0.8179	0.8239	0.8159	0.8241
SP3	CS	0.6423	0.7476	0.7524	0.7500	0.7574	0.7134	0.7584	0.7073
	GR	0.6423	0.7709	0.5726	0.7886	0.7173	0.7887	0.7982	0.8364
	IG	0.8421	0.8009	0.8428	0.8154	0.8440	0.8456	0.8494	0.8504
	RFF	0.8447	0.7709	0.8353	0.7577	0.8419	0.7722	0.8506	0.8414
	RFT	0.8447	0.7663	0.8429	0.7658	0.8419	0.7852	0.8439	0.8452
	SU	0.7663	0.7709	0.8336	0.7886	0.8416	0.8346	0.8397	0.8415
	KS	0.8331	0.7839	0.8308	0.7886	0.8515	0.8385	0.8531	0.8396
SP4	CS	0.6506	0.6817	0.7421	0.6913	0.7562	0.6666	0.7772	0.6697
	GR	0.6506	0.7837	0.6149	0.8153	0.7667	0.7872	0.8094	0.8177
	IG	0.8239	0.8113	0.8247	0.8094	0.8269	0.8317	0.8269	0.8323
	RFF	0.7895	0.7837	0.8069	0.7813	0.8077	0.7982	0.8158	0.8197
	RFT	0.7895	0.7905	0.7859	0.7570	0.8077	0.7920	0.8103	0.8218
	SU	0.7789	0.7837	0.7914	0.8153	0.8278	0.8290	0.8304	0.8342
	KS	0.7881	0.8112	0.7869	0.8217	0.8281	0.8199	0.8258	0.8208

Table 3. FRT vs. HFS – KNN

		Size 2		Size 3		Size 4		Size 6	
		FRT	HFS	FRT	HFS	FRT	HFS	FRT	HFS
SP2	CS	0.6552	0.6893	0.6798	0.6102	0.6989	0.6729	0.7922	0.7301
	GR	0.6552	0.7446	0.6634	0.7488	0.7825	0.7582	0.7804	0.7840
	IG	0.7140	0.7636	0.7735	0.7401	0.7655	0.7618	0.7327	0.7919
	RFF	0.7235	0.7446	0.7176	0.7128	0.7143	0.7362	0.7277	0.7662
	RFT	0.7235	0.7140	0.7110	0.6846	0.7143	0.7249	0.7891	0.7557
	SU	0.7534	0.7446	0.7735	0.7488	0.7794	0.7601	0.7740	0.7837
	KS	0.7729	0.7460	0.7743	0.7516	0.7727	0.7708	0.7717	0.7799
SP3	CS	0.6215	0.6500	0.6686	0.5497	0.6986	0.7277	0.6820	0.7606
	GR	0.6215	0.7233	0.6004	0.7447	0.7211	0.7529	0.7604	0.7445
	IG	0.7900	0.7436	0.7443	0.7577	0.7615	0.7652	0.7801	0.7867
	RFF	0.7485	0.7233	0.7458	0.7311	0.7269	0.8354	0.7770	0.7958
	RFT	0.7485	0.6706	0.7174	0.7288	0.7269	0.7602	0.7398	0.7774
	SU	0.6402	0.7233	0.7467	0.7447	0.7394	0.7329	0.7704	0.7467
	KS	0.7411	0.7092	0.7414	0.7486	0.7532	0.7269	0.7692	0.7516
SP4	CS	0.6394	0.5890	0.6444	0.5227	0.6684	0.5108	0.6993	0.6794
	GR	0.6394	0.7479	0.6275	0.7510	0.7543	0.7273	0.7602	0.7570
	IG	0.6784	0.7426	0.7804	0.7407	0.7796	0.7589	0.7893	0.7817
	RFF	0.7187	0.7479	0.6791	0.6940	0.6970	0.7086	0.7601	0.7943
	RFT	0.7187	0.6863	0.6787	0.6556	0.6970	0.7174	0.7677	0.7888
	SU	0.7447	0.7479	0.7454	0.7510	0.7833	0.7634	0.7623	0.7603
	KS	0.7541	0.7388	0.7374	0.7468	0.7741	0.7385	0.7845	0.7799

Table 4. FRT vs. HFS – SVM

		Size 2		Size 3		Size 4		Size 6	
		FRT	HFS	FRT	HFS	FRT	HFS	FRT	HFS
SP2	CS	0.6348	0.5000	0.7351	0.3002	0.7559	0.7480	0.6030	0.6837
	GR	0.6348	0.7432	0.4828	0.5951	0.6363	0.4784	0.7214	0.4890
	IG	0.6471	0.5041	0.7838	0.7722	0.7194	0.8004	0.6182	0.7934
	RFF	0.6844	0.7432	0.6520	0.5819	0.5967	0.5511	0.6689	0.7776
	RFT	0.6844	0.7269	0.7801	0.5169	0.5967	0.7520	0.5194	0.7031
	SU	0.5395	0.7432	0.7052	0.5951	0.8050	0.6902	0.2820	0.7697
	KS	0.7884	0.5968	0.6729	0.7000	0.8106	0.6902	0.8000	0.6166
SP3	CS	0.6321	0.4997	0.7484	0.2508	0.7502	0.7656	0.5415	0.6456
	GR	0.6321	0.7522	0.4701	0.6445	0.6152	0.5555	0.7613	0.4943
	IG	0.6763	0.5818	0.8283	0.7993	0.6200	0.8437	0.6969	0.7330
	RFF	0.7347	0.7522	0.6953	0.5291	0.6213	0.4463	0.7039	0.8465
	RFT	0.7347	0.7481	0.8267	0.4208	0.6213	0.7494	0.4278	0.7443
	SU	0.5352	0.7522	0.6766	0.6445	0.8340	0.6571	0.2763	0.8409
	KS	0.8293	0.5435	0.7251	0.6967	0.8253	0.6862	0.8373	0.6496
SP4	CS	0.6355	0.4999	0.7398	0.3590	0.7526	0.6985	0.6012	0.5900
	GR	0.6355	0.7612	0.4752	0.5845	0.6570	0.6664	0.6980	0.5795
	IG	0.6979	0.5460	0.7645	0.7992	0.6412	0.8027	0.6431	0.7319
	RFF	0.7475	0.7612	0.7126	0.5099	0.6206	0.3684	0.6414	0.7532
	RFT	0.7475	0.7385	0.7759	0.3490	0.6206	0.7451	0.4210	0.6649
	SU	0.5398	0.7612	0.6917	0.5845	0.8122	0.6642	0.2948	0.7608
	KS	0.7683	0.5709	0.6706	0.6662	0.7839	0.6504	0.8094	0.5472

Table 5. FRT vs. HFS – LR

		Size 2		Size 3		Size 4		Size 6	
		FRT	HFS	FRT	HFS	FRT	HFS	FRT	HFS
SP2	CS	0.6612	0.7458	0.7377	0.7324	0.7608	0.7282	0.7936	0.7386
	GR	0.6612	0.7701	0.7130	0.8054	0.8227	0.8113	0.8176	0.8294
	IG	0.8249	0.8103	0.8260	0.8170	0.8233	0.8338	0.8178	0.8310
	RFF	0.8077	0.7701	0.8121	0.7729	0.8129	0.8078	0.8180	0.8338
	RFT	0.8077	0.7641	0.8084	0.7687	0.8129	0.7930	0.8183	0.8294
	SU	0.8117	0.7701	0.8170	0.8054	0.8277	0.8295	0.8275	0.8367
	KS	0.8132	0.7968	0.8170	0.8247	0.8224	0.8291	0.8197	0.8336
SP3	CS	0.6423	0.7478	0.7529	0.7508	0.7552	0.7455	0.7561	0.7687
	GR	0.6423	0.7709	0.7155	0.7878	0.7875	0.7839	0.8163	0.8427
	IG	0.8441	0.7997	0.8433	0.8181	0.8474	0.8483	0.8529	0.8528
	RFF	0.8456	0.7709	0.8444	0.7606	0.8448	0.7824	0.8522	0.8425
	RFT	0.8456	0.7686	0.8470	0.7619	0.8448	0.7853	0.8481	0.8443
	SU	0.7653	0.7709	0.8327	0.7878	0.8425	0.8397	0.8421	0.8444
	KS	0.8335	0.7829	0.8315	0.8442	0.8484	0.8424	0.8477	0.8489
SP4	CS	0.6506	0.6819	0.7419	0.6958	0.7556	0.6993	0.7786	0.7259
	GR	0.6506	0.7836	0.6861	0.8160	0.7871	0.8050	0.8081	0.8265
	IG	0.8272	0.8085	0.8277	0.8125	0.8332	0.8358	0.8361	0.8374
	RFF	0.7873	0.7836	0.8102	0.7843	0.8094	0.8112	0.8182	0.8354
	RFT	0.7873	0.7937	0.7862	0.7465	0.8094	0.7977	0.8182	0.8295
	SU	0.7789	0.7836	0.7898	0.8160	0.8298	0.8343	0.8335	0.8357
	KS	0.7866	0.8119	0.7862	0.8270	0.8314	0.8271	0.8313	0.8303

expert concluded that for a high-dimensional software measurement data, such as LLTS, $\lceil \log_2 n \rceil$ (n is the number of available software metrics) adequately represents the upper limit for the number of selected attributes for defect prediction. The results presented in this section are generalization results, i.e., prediction for the test data sets.

We first used the seven filter-based feature ranking techniques to order the attributes based on their respective criteria. Then the top 2, 3, 4, or 6 features were selected from a given feature ranking. This reduced set of software metrics then forms the underlying training data for building the software quality prediction models. The algorithm for the filter-based feature ranking techniques and the subsequent classification modeling is presented in Figure 2. Similarly, the algorithm of our proposed hybrid feature selection method [16, 20] and the subsequent classification modeling is presented in Figure 3.

The HFS technique uses a feature subset selection search algorithm. To determine the attribute search space, the top 30% of the number of software attributes (i.e., 12 out of 42 features) were first extracted from each feature ranked list – this was once again based on the domain experts recommendation. In addition to reducing the search space for the AHS search algorithm, the top 12 attributes of a given ranking represents the 12 most relevant software metrics for fault-proneness prediction. From this top 12 attributes, different k features were selected using the AHS algorithm [20], where k was varied with values of 2, 3, 4, and 6. The classification performances were evaluated in terms of the performance metric, AUC. All the results are reported in Tables 1 through 5, one for each classifier.

In the tables, each value is determined by five dimensions: (1) feature selection technique (FRT vs. HFS); (2) number of selected attributes (2, 3, 4 and 6); (3) ranking technique (CS, GR, IG, RFF, RFT, SU and KS); (4) test datasets (SP2, SP3 and SP4); and (5) five classifiers (NB, MLP, KNN, SVM and LR). For example, the first value

```
for each ranking technique (CS, GR, IG, RFF, RFT, SU and KS)
  rank features using training data SP1
  select top k features from each ranked list (k=2, 3, 4 and 6)
  for each size k of feature subset
    for each classifier (NB, MLP, KNN, SVM, and LR)
      build classification models on SP1
      validate the model and collect the performance measure
        using test data SP2, SP3, and SP4
    end
  end
end
```

Figure 2. FRT experimental procedure

```
for each ranking technique (CS, GR, IG, RFF, RFT, SU and KS)
  rank features using training data set SP1
  select top 30% features (12 features) from each ranked list
  select size k of features using AHS (k=2, 3, 4, and 6)
  for each size k of feature subset
    for each classifier (NB, MLP, KNN, SVM, and LR)
      build classification models on SP1
      test the model and collect the performance metric using test
        data SP2, SP3, and SP4
    end
  end
end
```

Figure 3. HFS experimental procedure

in Table 1, 0.6612, refers to the predictive accuracy (AUC value) of the NB classifier built with two attributes on the first test dataset (SP2), where the two attributes were selected using the CS ranking technique. A total of 840 values are included in the five tables.

We conducted a three-way analysis of variance (ANOVA) F-test [2] on the performance metric, AUC, to statistically examine the various effects on the performances of the classification models. We concentrated on classification performances that were influenced by various feature selection techniques and different size (k) of the selected attribute subset. Hence, the ANOVA analysis is performed across all the five learners and across all three test datasets. The three ANOVA factors are: Factor A which represents the two different groups of feature selection techniques (FRT and HFS); Factor B which represents the four different sizes that were inspected for the attribute selection; and Factor C which represents the seven filter-based ranking techniques. The interaction effects of two factors, $A \times B$ and A$\times$C, were also considered in the ANOVA test. Note that a total of 56 subsets of attributes were used in the ANOVA test.

The three-way ANOVA test result is presented in Table 6. The p-values for the factors A and B, and the interaction terms A$\times$B and A$\times$C are less than a typical cutoff value of 0.05 – indicating the classification performances are not the same for all groups in each factor or term. In other words, the classification performances are significantly different from each other for at least a pair of groups in the corresponding factors or terms. For Factor A, the p-values (0.36) is much larger than the cutoff value of 0.05, which implies no significant difference exists between the two feature selection techniques (FRT vs. HFS) investigated in this study. However, the performance of the classification models based on HFS is slightly better than performance when feature ranking techniques are used alone.

Multiple comparisons were performed for the three main factors as well as the two interaction terms A$\times$B and A$\times$C

Table 6. Analysis of variance

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.005	1	0.005	0.8347	0.361
B	0.2199	3	0.0733	12.1249	0
C	0.8693	6	0.1449	23.9703	0
A$\times$B	0.0622	3	0.0207	3.4277	0.017
A$\times$C	0.1885	6	0.0314	5.1974	0
B$\times$C	0.239	18	0.0133	2.1969	0.003
A$\times$B$\times$C	0.4429	18	0.0246	4.0705	0
Error	4.7388	784	0.006		
Total	6.7655	839			

to identify which pair(s) of means significantly differ from each other in each factor or term. The test results are shown in Figure 4, Each sub-figure displays graphs with each group mean represented by a symbol ($\circ$) and an interval around the symbol (95% confidence interval). Two means are significantly different (α = 0.05) if their intervals are disjoint, and are not significantly different if their intervals overlap. The following conclusions are made:

1. For Factor A (feature selection technique), FRT and HFS performed very similar with HFS showing slightly better performance than FRT. This is consistent with the conclusion obtained from ANOVA test.

2. For Factor B (size of attribute subset), we can group the four different k sizes into two classes. Class1 includes size 2 and size 3, and Class2 include size 4 and size 6. Figure 4 (b) shows that the classification models built with 4 or 6 attributes (Class2) significantly outperformed the classification models built with 2 or 3 attributes (Class1). The different sizes in the same class are very similar to each other, implying that the subset of 4 metrics provides similar performance as the subset of 6 metrics (i.e., $\lceil \log_2 n \rceil$)

3. For Factor C (ranking techniques), CS performed significantly worse than other techniques, and GR is better than CS but worse than the others. IG and KS

Table 7. Classification on 42 attributes

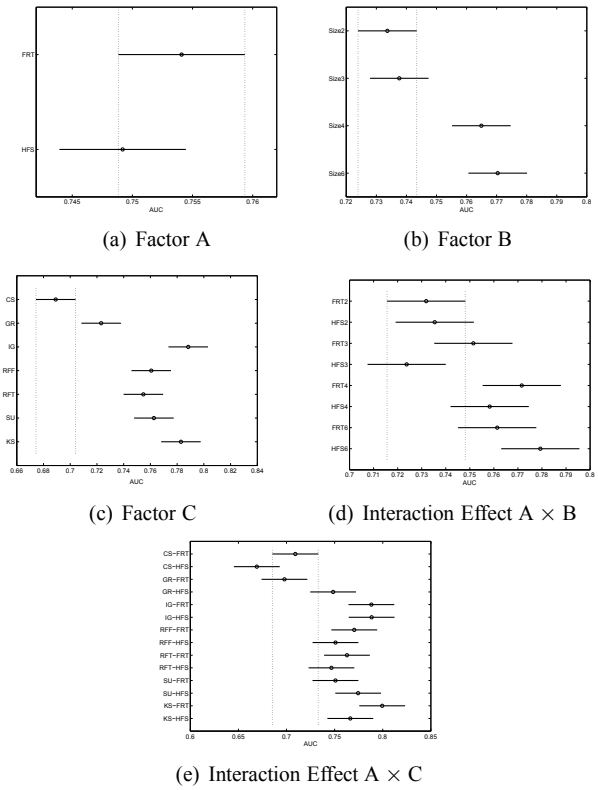
Classifier	SP2	SP3	SP4
NB	0.8149	0.7963	0.8059
MLP	0.8314	0.8322	0.8309
KNN	0.7849	0.8054	0.7901
SVM	0.6662	0.6519	0.6779
LR	0.8287	0.7989	0.8246

showed the best performances in this study, while RFT, RFF, and SU demonstrated similar performances.

4. For the interaction terms A×B, there are eight categories (levels). Figure 4 (d) shows that when using the two feature selection techniques (FRT and HFS) to select 4 or 6 attributes, we can obtain better prediction from the classification models. The 4-attribute subset selected directly by feature rankings provides very good results.

5. For the interaction terms A×C, there are 14 categories (levels), which represents the 14 different attribute subset selection techniques we used in the study (seven direct rankings plus seven hybrid subset selections). Figure 4 (e) demonstrates that CS performed poorly regardless of whether it was used alone or in combination with the AHS search algorithm. While the GR ranking technique provided poor prediction when used alone, its performance improves when combined with the AHS search algorithm. All the other techniques showed no significant difference in terms of their performances. However, the classification performances based on the subsets of attributes selected with IG-FRT (IG used alone), IG-HFS (IG combined with HFS), and KS-FRT (KS used alone) methods, were generally better than those of others.

Referring to the research question posed earlier in the paper, our results have demonstrated that for the LLTS system when 4 out of 42 attributes were used to train the classification models, we obtain the same generalization performances as compared to when a larger subset of metrics (i.e., six attributes) or when the complete set of attributes (42 attributes) is used for training. Results of the latter are shown in Table 7.



(a) Factor A (b) Factor B

(c) Factor C (d) Interaction Effect A × B

(e) Interaction Effect A × C

Figure 4. Multiple comparisons

6 Conclusion

The paper addresses the question of "what is the minimum number of software metrics that should be used to build a software quality prediction model for a given system?" A detailed study of seven filter-based feature ranking techniques and our proposed hybrid feature selection technique is presented in the context of an empirical software engineering study. Software measurement data from four releases of large telecommunications system is used in our case study. Software quality prediction models are built using five different classification algorithms: including naive Bayes, multilayer perceptron, K-nearest neighbors, support vector machine, and logistic regression.

The answer to the above stated question is that for the LLTS system, only 10% of the available software metrics were sufficient in building a useful software quality prediction model. This implies that only four out of the 42 software metrics are considered useful – this result is verified by a software engineering domain expert with over 20 years of experience in software quality engineering. Another conclusion is that the Kolmogorov-Smirnov technique for feature ranking provided competitive performance as compared to the other approaches. Finally, our proposed hybrid feature selection technique also performed better when 4 software attributes are selected.

Future work will focus on experimental analysis of other software project, especially those from other application domains. From a practical point of view, an analysis on which software metrics stand out as good attributes for defect prediction across multiple projects would be of interest to the software engineering community.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):1573–0565, January 1991.

[2] M. L. Berenson, M. Goldstein, and D. Levine. *Intermediate Statistical Methods and Applications: A Computer Package Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2 edition, 1983.

[3] A. C. Cameron and P. K. Trivedi. *Regression Analysis of Count Data*. Cambridge University Press, 1998.

[4] S. Doraisamy, S. Golzari, N. M. Norowi, N. Sulaiman, and N. I. Udzir. A study on feature selection and classification techniques for automatic genre classification of traditional malay music. In *Ninth International Conference on Music Information Retrieval*, pages 331–336, Philadelphia, PA, USA, Sept. 14-18 2008.

[5] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[6] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Company: ITP, Boston, MA, 2nd edition, 1997.

[7] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, March 2003.

[8] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[9] M. A. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1437 – 1447, Nov/Dec 2003.

[10] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 2 edition, 1998.

[11] J. P. Hudepohl, S. J. Aud, T. M. Khoshgoftaar, E. B. Allen, and J. Mayrand. EMERALD: Software metrics and models on the desktop. *IEEE Software*, 13(5):56–60, September 1996.

[12] G. Ilczuk, R. Mlynarski, W. Kargul, and A. Wakulicz-Deja. New feature selection methods for qualification of the patients for cardiac pacemaker implantation. In *Computers in Cardiology, 2007*, pages 423–426, Durham, NC, USA, 2007.

[13] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, volume 2, pages 338–345, San Mateo, 1995.

[14] K. Jong, E. Marchiori, M. Sebag, and A. van der Vaart. Feature selection in proteomic pattern data with support vector machines. In *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, Oct 7-8 2004.

[15] T. M. Khoshgoftaar, L. A. Bullard, and K. Gao. Attribute selection using rough sets in software quality classification. *International Journal of Reliability, Quality and Safty Engineering*, 16(1):73–89, 2009.

[16] T. M. Khoshgoftaar, L. Nguyen, K. Gao, and J. Rajeevalochanam. Application of an attribute selection method to cbr-based software quality classification. In *Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 47–52, Sacramento, California, November 3-5 2003. IEEE.

[17] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of 9th International Workshop on Machine Learning*, pages 249–256, 1992.

[18] S. Le Cessie and J. C. Van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[19] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4):485–496, July-August 2008.

[20] P. Lin, H. Wang, and T. M. Khoshgoftaar. A novel hybrid search algorithm for feature selection. In *Proceedings of 21st International Conference on Software Engineering and Knowledge Engineering*, pages 81–86, Boston, MA, July 1-3 2009.

[21] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005.

[22] M. J. Meulen and M. A. Revilla. Correlations between internal software metrics and software dependability in a large population of small C/C++ programs. In *Proceedings of the 18th IEEE International Symposium on Software Reliability Engineering, ISSRE 2007*, pages 203–208, Trollhattan, Sweden, November 2007.

[23] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, Los Altos, California, 1993.

[24] D. Rodriguez, R. Ruiz, J. Cuadrado-Gallego, and J. Aguilar-Ruiz. Detecting fault modules applying feature selection to classifiers. In *Proceedings of 8th IEEE International Conference on Information Reuse and Integration*, pages 667–672, Las Vegas, Nevada, August 13-15 2007.

[25] D. Rodríguez, R. Ruiz, J. Cuadrado-Gallego, J. Aguilar-Ruiz, and M. Garre. Attribute selection in software engineering datasets for detecting fault modules. In *Proceedings of 33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 418–423, Lübeck, Germany, August 27-31 2007.

[26] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines*. Cambridge University Press, 2 edition, 2000.

[27] K. Sunghun, T. Zimmermann, E. J. Whitehead, and A. Zeller. Predicting faults from cached history. In *Proceedings of the 29th International Conference on Software Engineering, ICSE 2007*, pages 489–498, 2007.

[28] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2 edition, 2005.