December 2009

Mining Data from Multiple Software Development Projects

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

Mining Data from Multiple Software Development Projects

Huanjing Wang, Taghi M. Khoshgoftaar, Kehan Gao, and Naeem Seliya
{huanjing.wang@wku.edu; taghi@cse.fau.edu; gaok@easternct.edu; nseliya@umich.edu}

Abstract

A large system often goes through multiple software project development cycles, in part due to changes in operation and development environments. For example, rapid turnover of the development team between releases can influence software quality, making it important to mine software project data over multiple system releases when building defect predictors. Data collection of software attributes are often conducted independent of the quality improvement goals, leading to the availability of a large number of attributes for analysis. Given the problems associated with variations in development process, data collection, and quality goals from one release to another emphasizes the importance of selecting a best-set of software attributes for software quality prediction. Moreover, it is intuitive to remove attributes that do not add to, or have an adverse effect on, the knowledge of the consequent model. Based on real-world software projects' data, we present a large case study that compares wrapper-based feature ranking techniques (WRT) and our proposed hybrid feature selection (HFS) technique. The comparison is done using both three-fold cross-validation (CV) and three-fold cross-validation with risk impact (CVR). It is shown that HFS is better than WRT, while CV is superior to CVR.

***Keywords**: data preparation, attribute selection, data selection, software measurements, defect prediction.*

1 Introduction

Developing high-quality software is a primary goal for any development team. Software practitioners strive to improve software quality by constructing software quality prediction models [9]. However, when a large system goes through multiple software releases, it is very likely that the characteristics and trends of software quality change from one project development to another. A given system release can have relatively different operation and development environments, which would be reflected in the software development process. Moreover, a rapid turnover of team members can also influence software quality characteristics. Thus, it is important to analyze software data from multiple sources, i.e., multiple development projects or releases, when building software quality prediction models.

A typical defect prediction model is built using software metrics and fault data of previously developed releases or similar projects. Then the quality of the under-development program modules is estimated, as fault-prone (*fp*) or not fault-prone (*nfp*), for example. As described earlier, characteristics of the software measurement data set play an important role in the efficacy of the defect prediction models. One such aspect is which software attributes are good quality indicators for the given system, especially when there is a likelihood of variation in development processes, data collection methods and objectives, and quality improvement goals from one release to another. We focus on feature selection in the context of software quality prediction and analysis when software measurement and defect data from multiple development sources/releases is available.

Feature selection is an essential task during the data preparation and data cleansing phases of a machine learning problem [12]. The aim of feature selection is to find a feature subset that characterizes the underlying data as good as, or even better, than the original data when all available features are considered. *Feature ranking* techniques rank the attributes based on their individual predictive power, while *feature selection* approaches select subsets of attributes that collectively have good predictive power. Feature subset selection can be categorized as *filters*, *wrappers*, or *embedded* methods. Filters are algorithms in which a feature subset is selected without involving any learning algorithm. Wrappers are algorithms that use feedback from a learning algorithm to determine which feature(s) to include for building a classifier. Embedded methods imply that the feature selection aspect is incorporated within a learning algorithm.

We evaluate two types of feature selection methods: *wrapper-based feature ranking techniques* (WRT) and our proposed *hybrid feature selection method* (HFS). In the case of WRT, classifiers with a single independent attribute (and the dependent class attribute) are built, and then the attributes are selected according to their individual predictive capability as measured by a performance metric. We used five different and commonly used learners [14]: Naïve Bayes (NB), Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Logistic Regression (LR). In addition, five different performance metrics are used: Overall Accuracy (OA), Default Geometric Mean (DGM), Area Under ROC (AUC), Area Under PRC (PRC) and Best Arithmetic Mean (BAM). In total, we had 25 rankers – based on five learners and five performance metrics. An unique contribution of our work is that we build classifiers using 25 rankers.

The proposed HFS technique consists of a wrapper-based feature ranking technique and a consistency-based feature subset selection – the latter is our proposed automatic hybrid search (AHS) [10]. We assessed the feature selection methods (WRT and HFS) by building classification models with the selected attributes and evaluating them with two performance metrics [14]: Area Under the ROC (receiver operator characteristic) curve (AUC) and Best Geometric Mean (BGM). When building a defect prediction classifier, we considered two model-building strategies: three-fold cross-validation (CV) and three-fold cross-validation risk impact (CVR). For each strategy, features were selected through WRT and HFS, and we analyze which strategy provides better performance for a given classifier. Such a large comparative study of feature selection for defect prediction modeling is relatively unique to this paper. Moreover, all data sets analyzed in this study suffer from class imbalance, and investigating feature selection with such data sets poses additional challenges.

The paper is organized as follows. Section 2 provides information on the cross-validation strategies, feature selection techniques, and learners used. Section 3 provides a description of the software projects' data, and Section 4 details the empirical results. Relevant literature on feature selection is presented in Section 5. Section 6 concludes the paper and provides suggestions for future work.

2 Methodology

2.1 Cross-Validation Methods

We use two strategies for the WRT approaches. First, a three-fold cross-validation (CV) method is used. For each attribute to be ranked, a reduced dataset consisting solely of that attribute and the class attribute is created. For each of the three folds (partitions), a classifier is trained on the other two folds (partitions), then tested on that fold. For each learner, all five performance metrics are calculated based on cross-validation. Performance scores are obtained in this manner for all attributes in the data set, one at a time. After the five performance metrics are obtained for all five learners for every attribute, features are ranked based on a given performance metric for each classifier.

The second method, three-fold cross-validation with risk impact (CVR), is similar to CV except that each instance of the training data set contains $n-1$ independent features and the class feature. For each learner, we build a model with all n independent features and the class feature and recording all performance measures, denoted as PM_n, where PM represents one of the five performance metrics. For each independent feature i, we eliminate it, and build a model with one less independent feature to get the associated performance measure, denoted as PM^i_{n-1}, where $1 \leq i \leq n$. The risk impact of each feature i for a particular learner is defined as $PM_n - PM^i_{n-1}$. Each feature has a risk impact score and these features are ranked in a descending order according to risk impact scores. CVR is more time consuming than CV since it builds models using $n-1$ independent features while CV uses only one independent feature.

2.2 Wrapper-Based Ranking Techniques

The WRT feature selection approach was implemented by our team within the WEKA framework [14]. The approach consists of two parts, a learner (classifier) and a performance metric, that collectively form a ranker (Figure 1). In the feature ranking technique, every feature in a given fit (training) data set is used individually in the model building process. We used the CV and CVR strategies to build the classification models and then assessed their performances based on a specific performance metric. We then ranked the features and selected the top $\lceil \log_2 n \rceil$ features according to their respective scores, where n is number of independent features for a given data set. In this study, we selected six features from the attribute set that contained 42 features.

The reasons for selecting the top $\lceil \log_2 n \rceil$ features are: (1) no general guidance is available on the number of features that should be selected when using a feature ranking technique; (2) in a recent empirical study [8], we showed that it was appropriate to use $\lceil \log_2 n \rceil$ as the number of features when using WEKA [14] to build a classifier for binary classification in general, and for imbalanced data sets in particular. Thus, we assume $\lceil \log_2 n \rceil$ is a good choice for this research; and (3) a software engineering expert with more than 20 years experience recommends selecting $\lceil \log_2 n \rceil$ number of metrics for high-assurance systems such as our case study system.

In a two-group classification, e.g., *fp* and *nfp*, there are four possible outcomes: true positive (TP), false positive (FP), true negative (TN) and false negative (FN), where positive represents *fp* and negative represents *nfp*. The respective frequency of the four outcomes form the basis for several other performance measures that are commonly used for classifier evaluation. We use five of them for the wrapper-based feature ranking process, and they are:

1. Overall Accuracy (OA) provides a single value range

from 0 to 1. It can be obtained by $\frac{|TP|+|FN|}{N}$, where N is the total number of instances in the data set.

2. Default Geometric Mean (DGM) is a single-value measure that ranges from 0 to 1, and a perfect classifier provides a value of 1. Geometric Mean (GM) is defined as the square root of the product of true positive rate(defined as $\frac{|TP|}{|TP|+|FN|}$) and true negative rate (defined as $\frac{|TN|}{|FP|+|TN|}$). The threshold, $t = 0.5$, is used for DGM, while the *Best Geometric Mean* (BGM) is the maximum GM value that is obtained when varying the threshold between 0 and 1.

3. Area Under the ROC Curve (AUC) has been widely used to measure classification model performance [2]. The ROC (Receiver Operating Characteristic) curve characterizes the trade-off between the true positive rate and the false positive rate. A perfect classifier provides an AUC that equals 1.

4. Area Under the Precision-Recall Curve (PRC) is a single-value measure that depicts the trade off between Recall and Precision [14]. A classifier that is optimal in AUC space may not be optimal in PRC space.

5. Best Arithmetic Mean (BAM) is based on the arithmetic mean of the true positive rate and true negative rate values. The *Best Arithmetic Mean* (BAM) uses the maximum arithmetic mean that is obtained when varying the threshold between 0 and 1.

2.3 Proposed Hybrid Feature Selection Method

Feature subset selection techniques search the set of possible features as a group and evaluate their collective suitability. However, the traditional approaches to feature selection with single evaluation criterion have shown limited capability in terms of knowledge discovery and decision support. We proposed a hybrid feature selection (HFS) method which combines wrapper ranking and the Automatic Hybrid Search (AHS), our recently proposed feature subset selection method [10]. AHS uses the consistency rate properties, relying on its monotonic property of consistency rate. The AHS algorithm was developed and implemented by our team in Java.

The AHS algorithm works as follows. The consistency rate of the full feature set is computed first, and then starting with size 1 of any feature, the feature subsets that have the locally highest consistency rate are selected. These selected feature 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 features is reached. The newly proposed hybrid feature selection (HFS) method consists of two steps (Figure 2). First, use a wrapper-based feature ranking technique to rank the features and the top 30% of features are selected from the ranked list. Twelve features are selected in this study; Second, apply AHS to select a subset of k features with the highest local consistency rate. The size of feature subset depends on the application domain and project characteristics. In this study, k is set to six.

2.4 Classifiers

Classification is a form of data analysis that can be used to extract a model that minimizes the number of classification errors on a training data set. The first step of classification is to build a classification model that can describe the predetermined set of data classes, whereas in the second step, the classification model is evaluated using an independent test data set. The software quality prediction models are built with five different learners available in WEKA [14], including: Naïve Bayes, Multilayer Perceptron, K-Nearest Neighbors, Support Vector Machines, and Logistic Regression. These learners are used in data mining serve as both aids to the ranking process and an inductive learner for the classification performance process. Unless stated otherwise, we use default parameter settings for the different learners as specified in WEKA [14]. Parameter settings are changed only when a significant improvement in performance is obtained.

3 Software Measurement Data Description

The software metrics and defect data used in our case study were collected from four consecutive releases of a very large telecommunications software system (denoted as LLTS). Each of the software measurement data sets consist of 42 software metrics: 24 product, 14 process, and four execution metrics [5]. Details on these metrics are avoided due to paper size consideration. A program module with one or more faults is considered *fp*, and *nfp* otherwise. Labeled chronologically as SP1, SP2, SP3 and SP4, the four data sets consist of 3649, 3981, 3541 and 3978 modules, respectively. These data sets suffer from the class imbalance problem, i.e., the proportion of *fp* modules is much lower than that of the *nfp* modules. The proportions of *nfp* modules of four releases are 93.72%, 95.25%, 98.67% and 97.69%, respectively. The SP1 data set is used as fit (training) data, while the SP2, SP3 and SP4 data sets are used as independent test (evaluation) data.

4 Empirical Case Study and Results

The experiments were conducted to discover the impact of (1) standard cross-validation vs. cross-validation risk impact; (2) wrapper based feature ranking vs. hybrid feature

```
for each cross-validation method (CV and CVR)
  for  each learner (NB, MLP, KNN, SVM, and LR)
    for  each ranking technique (OA, DGM, AUC, PRC, and BAM)
      rank features using fit data SP1
      select top 6 features from ranked list
      build classification model using reduced data set on SP1 with
         same learner
      validate the model and collect the performance metrics AUC
         and BGM using test data SP2, SP3 and SP4
    end
  end
end
```

Figure 1. WRT experimental procedure

```
for each cross-validation method (CV and CVR)
  for  each learner (NB, MLP, KNN, SVM, and LR)
    for  each ranking technique (OA, DGM, AUC, PRC, and BAM)
      rank features using fit data SP1
      select top 30% features (12 features) from ranked list
      select size 6 of feature subset using AHS
      build classification model using reduced data set on SP1 with
         same learner
      validate the model and collect the performance metrics AUC
         and BGM using test data SP2, SP3 and SP4
    end
  end
end
```

Figure 2. HFS experimental procedure

selection; (3) five different performance metrics used for the rankers; and (4) five different learners.

We first used wrapper-based feature ranking to select the subsets of attributes. Two approaches (CV and CVR), five learners (NB, MLP, KNN, SVM and LR), and five performance metrics (OA, DGM, AUC, PRC and BAM) form the basis of our wrapper-based ranking techniques. The algorithm for the wrapper-based feature ranking techniques and the subsequent classification modeling is presented in Figure 1. We then used our proposed HFS technique to select the subsets of attributes. The HFS technique is a combination of wrapper-based feature ranking and a (our proposed) feature subset selection algorithm, AHS. The algorithm of the HFS technique and the subsequent defect prediction modeling is presented in Figure 2.

The defect predictors were evaluated in terms of AUC and BGM. The results are summarized in Tables 1 through 5, where each value in a given table is based on six dimensions: (1) feature selection technique (WRT vs. HFS); (2) performance metric for feature ranking (OA, DGM, AUC, PRC or BAM); (3) cross-validation strategy (CV vs. CVR); (4) five classifiers (NB, MLP, KNN, SVM or LR); (5) performance metric for the final models (AUC or BGM); and

Table 1. Learner – NB

		CV				CVR			
		AUC		BGM		AUC		BGM	
		WRT	HFS	WRT	HFS	WRT	HFS	WRT	HFS
	OA	0.6977	0.7521	0.6552	0.7027	0.7653	0.7470	0.7190	0.6985
	DGM	0.7697	0.8124	0.7226	0.7543	0.8246	0.7442	0.7608	0.7046
SP2	AUC	0.8217	0.8241	0.7576	0.7583	0.8272	0.8274	0.7560	0.7554
	PRC	0.8151	0.8151	0.7474	0.7565	0.8228	0.8148	0.7489	0.7635
	BAM	0.8192	0.8205	0.7552	0.7637	0.8188	0.7660	0.7555	0.7088
	OA	0.6629	0.7442	0.6618	0.7145	0.7616	0.7471	0.7194	0.7248
	DGM	0.8011	0.8250	0.7686	0.7821	0.8036	0.7390	0.7293	0.7026
SP3	AUC	0.8158	0.8088	0.7709	0.7518	0.8189	0.8259	0.7759	0.7691
	PRC	0.8352	0.8314	0.7952	0.7884	0.8236	0.7962	0.7635	0.7660
	BAM	0.8132	0.8142	0.7481	0.7698	0.8294	0.7678	0.7624	0.7337
	OA	0.6710	0.7415	0.6817	0.6966	0.7745	0.7283	0.7380	0.7024
	DGM	0.7791	0.8093	0.7449	0.7383	0.8006	0.7682	0.7396	0.7279
SP4	AUC	0.8132	0.8173	0.7316	0.7264	0.8081	0.8276	0.7227	0.7370
	PRC	0.8264	0.8125	0.7612	0.7347	0.8038	0.7826	0.7221	0.7293
	BAM	0.8136	0.8117	0.7441	0.7298	0.8211	0.7671	0.7557	0.7256

Table 2. Learner – MLP

		CV				CVR			
		AUC		BGM		AUC		BGM	
		WRT	HFS	WRT	HFS	WRT	HFS	WRT	HFS
	OA	0.7687	0.8285	0.7275	0.7552	0.8055	0.8290	0.7384	0.7607
	DGM	0.7922	0.8285	0.7339	0.7552	0.7802	0.7921	0.7208	0.7309
SP2	AUC	0.8093	0.8055	0.7444	0.7473	0.8329	0.8252	0.7600	0.7465
	PRC	0.8299	0.8304	0.7617	0.7636	0.8038	0.8300	0.7416	0.7531
	BAM	0.7978	0.8051	0.7430	0.7476	0.8295	0.8311	0.7509	0.7563
	OA	0.7449	0.8372	0.7183	0.7875	0.7916	0.8499	0.7600	0.8128
	DGM	0.7582	0.8372	0.7226	0.7875	0.7321	0.7947	0.7363	0.7552
SP3	AUC	0.8305	0.8331	0.7761	0.7750	0.8484	0.8427	0.7921	0.7730
	PRC	0.8514	0.8467	0.8080	0.7886	0.8329	0.8418	0.8123	0.7815
	BAM	0.8329	0.8325	0.7728	0.7754	0.8433	0.8440	0.7802	0.7815
	OA	0.7689	0.8268	0.7620	0.7508	0.7848	0.8377	0.7202	0.7691
	DGM	0.7650	0.8268	0.7565	0.7508	0.7724	0.7725	0.7076	0.7140
SP4	AUC	0.7899	0.7881	0.7195	0.7178	0.8248	0.8237	0.7469	0.753
	PRC	0.8318	0.8274	0.7585	0.7539	0.7895	0.8220	0.7225	0.7525
	BAM	0.7849	0.7886	0.7190	0.7180	0.8296	0.8325	0.7511	0.7483

Table 3. Learner – KNN

		CV				CVR			
		AUC		BGM		AUC		BGM	
		WRT	HFS	WRT	HFS	WRT	HFS	WRT	HFS
	OA	0.5925	0.7118	0.6201	0.6490	0.7285	0.7210	0.6989	0.6797
	DGM	0.7779	0.7764	0.7197	0.7272	0.7363	0.7302	0.6997	0.6797
SP2	AUC	0.7684	0.7822	0.7192	0.7327	0.7534	0.7321	0.7119	0.6843
	PRC	0.7709	0.7722	0.7145	0.7265	0.7664	0.7608	0.7063	0.7096
	BAM	0.7772	0.7842	0.7255	0.7329	0.7534	0.7477	0.7119	0.6942
	OA	0.7226	0.7249	0.7210	0.6721	0.7397	0.7371	0.7059	0.7019
	DGM	0.7565	0.7808	0.7269	0.7595	0.7702	0.7637	0.7381	0.7367
SP3	AUC	0.7639	0.8124	0.7372	0.7790	0.7600	0.7510	0.7121	0.7388
	PRC	0.7739	0.8029	0.7189	0.7688	0.7588	0.7309	0.7102	0.7181
	BAM	0.7709	0.8178	0.7439	0.7952	0.7600	0.7764	0.7121	0.7493
	OA	0.6705	0.7534	0.7029	0.6798	0.6931	0.7119	0.6790	0.6957
	DGM	0.7610	0.7479	0.7105	0.6936	0.7621	0.7572	0.7101	0.7179
SP4	AUC	0.7535	0.7466	0.6797	0.6958	0.7495	0.7577	0.6962	0.6999
	PRC	0.7817	0.7439	0.7148	0.6856	0.7359	0.7358	0.6735	0.6971
	BAM	0.7666	0.7450	0.6940	0.6929	0.7495	0.7565	0.6962	0.7326

Table 4. Learner – SVM

		CV				CVR			
		AUC		BGM		AUC		BGM	
		WRT	HFS	WRT	HFS	WRT	HFS	WRT	HFS
	OA	0.6265	0.6516	0.5890	0.6015	0.7626	0.6317	0.7024	0.6645
	DGM	0.3798	0.3930	0.4326	0.4283	0.7546	0.7237	0.7111	0.6939
SP2	AUC	0.7187	0.4423	0.6964	0.4862	0.6300	0.7520	0.6208	0.6915
	PRC	0.7187	0.8123	0.6964	0.7432	0.7670	0.6278	0.6983	0.6050
	BAM	0.7997	0.8347	0.7253	0.7722	0.3715	0.4815	0.4333	0.5350
	OA	0.6114	0.6194	0.5752	0.5990	0.7360	0.7108	0.7031	0.6923
	DGM	0.4025	0.4179	0.4418	0.4674	0.7219	0.6651	0.6969	0.6636
SP3	AUC	0.6902	0.5761	0.6544	0.5968	0.6892	0.7611	0.6695	0.6954
	PRC	0.6902	0.8040	0.6544	0.7743	0.7613	0.6980	0.7460	0.6593
	BAM	0.8352	0.8351	0.7801	0.7771	0.3538	0.5391	0.4194	0.5538
	OA	0.5510	0.6078	0.5559	0.6161	0.7454	0.6543	0.6923	0.6792
	DGM	0.4725	0.5046	0.5290	0.5520	0.6939	0.7243	0.6794	0.6932
SP4	AUC	0.6996	0.5761	0.6704	0.5968	0.7561	0.7610	0.7059	0.7094
	PRC	0.6996	0.7881	0.6704	0.7243	0.7705	0.7510	0.7187	0.7186
	BAM	0.7683	0.8033	0.7162	0.7314	0.2913	0.3798	0.3730	0.4670

Table 5. Learner – LR

		CV				CVR			
		AUC		BGM		AUC		BGM	
		WRT	HFS	WRT	HFS	WRT	HFS	WRT	HFS
	OA	0.8258	0.8272	0.7443	0.7545	0.7648	0.7611	0.7092	0.6992
	DGM	0.8024	0.8263	0.7413	0.7482	0.7676	0.8033	0.7095	0.7462
SP2	AUC	0.8331	0.8310	0.7675	0.7681	0.8224	0.8224	0.7528	0.7528
	PRC	0.8331	0.8314	0.7675	0.7649	0.7626	0.8133	0.7151	0.7390
	BAM	0.8234	0.8333	0.7561	0.7634	0.8260	0.8234	0.7539	0.7556
	OA	0.8252	0.8468	0.7599	0.7881	0.7721	0.7572	0.7168	0.7017
	DGM	0.7840	0.8435	0.7232	0.7811	0.7592	0.7936	0.7131	0.7620
SP3	AUC	0.8471	0.8528	0.8040	0.8126	0.8374	0.8374	0.7937	0.7937
	PRC	0.8471	0.8527	0.8040	0.8122	0.7529	0.8053	0.7117	0.7572
	BAM	0.8473	0.8467	0.7915	0.7875	0.8485	0.8520	0.7915	0.8034
	OA	0.8315	0.8211	0.7711	0.7521	0.7416	0.7885	0.7073	0.7538
	DGM	0.8145	0.8377	0.7569	0.7626	0.7440	0.7826	0.7025	0.7281
SP4	AUC	0.8341	0.8374	0.7547	0.7595	0.8024	0.8024	0.7381	0.7381
	PRC	0.8341	0.8318	0.7547	0.7579	0.7473	0.7916	0.7388	0.7437
	BAM	0.8311	0.8285	0.7591	0.7479	0.8115	0.8103	0.7379	0.7395

Table 6. ANOVA in terms of AUC

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.1238	4	0.0309	8.38	0
B	0.0034	1	0.0034	0.93	0.336
C	1.1241	4	0.2810	76.11	0
D	0.0049	1	0.0049	1.32	0.252
A×B	0.1526	4	0.0382	10.33	0
A×C	0.1916	16	0.0120	3.24	0
A×D	0.0067	4	0.0017	0.46	0.768
B×C	0.0230	4	0.0058	1.56	0.186
B×D	0.0033	1	0.0033	0.90	0.345
C×D	0.0067	4	0.0017	0.46	0.768
Error	0.9453	256	0.0037		
Total	2.5856	299			

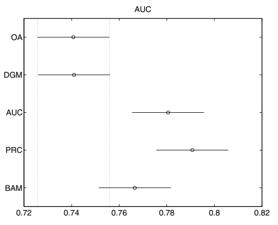
(6) test data set (SP2, SP3 or SP4). For example, the first value in Table 1, 0.6977, refers to the predictive accuracy in terms of AUC for the NB classifier on the first test data set (SP2), where the NB classifier was built with the six features that were selected using the ranker, which was formed by the NB learner in conjunction with the OA performance metric and using the CV approach on the fit data.

We perform a four-way ANalysis Of VAriance (ANOVA) F-test with respect to the AUC and BGM metrics (separately), to statistically examine the various effects on the models' performances. The four *main factors* in the ANOVA test are: Factor A, for the five performance metrics used for the rankings (OA, DGM, AUC, PRC and BAM); Factor B, for the two cross-validation strategies (CV and CVR); Factor C, for the five learners (NB, MLP, KNN, SVM and LR); and Factor D, for the two feature selection methods (WRT and HFS). The *interaction effects* of any two factors were also considered in the ANOVA test.

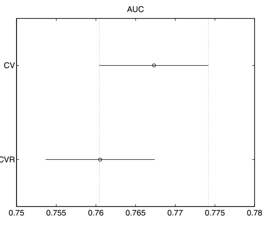
The ANOVA model can be used to test the hypothesis

Table 7. Classification on 42 attributes

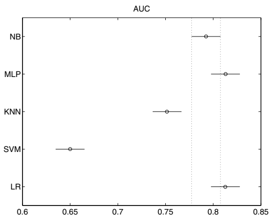
	SP2		SP3		SP4	
learner	AUC	BGM	AUC	BGM	AUC	BGM
NB	0.8149	0.7457	0.7963	0.7248	0.8059	0.7411
MLP	0.8314	0.7595	0.8322	0.7688	0.8309	0.7655
KNN	0.7849	0.7255	0.8054	0.7746	0.7901	0.7285
SVM	0.6662	0.6655	0.6519	0.6622	0.6779	0.6948
LR	0.8287	0.7486	0.7989	0.7557	0.8246	0.7382



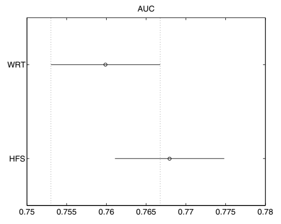
(a) Factor A



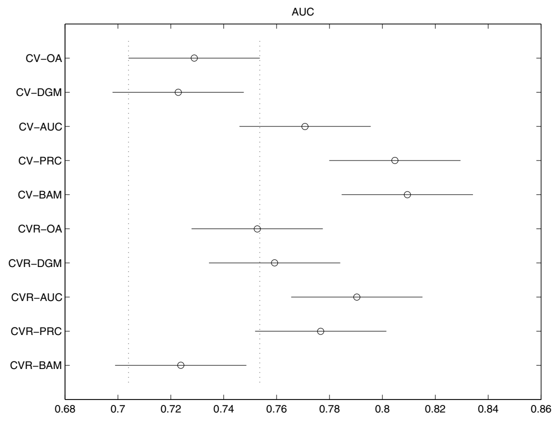
(b) Factor B



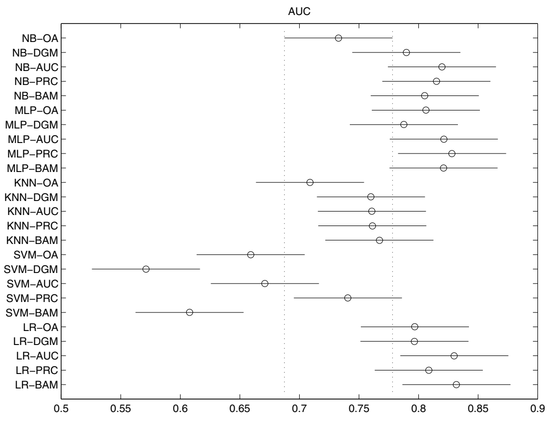
(c) Factor C



(d) Factor D



(e) Interaction Term A×B



(f) Interaction Term A×C

Figure 3. Multiple comparisons for AUC

that the AUC (or BGM) for the main factors A, B, C and D and/or for the interaction effects (terms) A×B, A×C, A×D, B×C, B×D and C×D are equal against the alternative hypothesis that at least one mean is different. If the alternative hypothesis is accepted, multiple comparisons can be used to determine which of the means are significantly different from the others. We performed the multiple comparison tests using Tukey's honestly significant difference (HSD) criterion. A significance level of $\alpha = 5\%$ is used for all statistical tests. The ANOVA results based on AUC are presented in Table 6. From the table, we can see that the p-values (last column of Table 6) for the main factors A and C and the interaction terms A×B and A×C are less than 0.05, indicating the AUC values are not similar for all groups in each of these factors or interaction terms.

Additional multiple comparisons for the main factors and interaction terms were performed to investigate the difference among the respective groups (levels). Although the ANOVA tests showed that all the group means are equal (similar) for Factor B and Factor D, we also performed the multiple comparisons for both factors to identify which group means are higher. The test results are shown in Figure 3, where each sub-figure displays graphs with each group mean represented by a symbol (○) and 95% confidence interval. The summarized results reveal the following:

1. For Factor A (Figure 3(a)), the rankers based on AUC or PRC as performance metrics significantly outperformed those using OA or DGM as performance metrics. The rankers based on the BAM performance metric come in within the two above groups.

2. For Factor B (Figure 3(b)), CV performed generally better than CVR (at significance level of 0.336). However, since CV has lower relative complexity, we recommend CV in this research.

3. For Factor C (Figure 3(c)), the NB, MLP and LR classifiers performed significantly better than the SVM and KNN classifiers. While MLP and LR performed the best, SVM performed the worst.

4. For Factor D (Figure 3(d)), HFS outperformed WRT (at significance level of 0.252). HFS searches a subset of features that collectively has good predictive power, while WRT only considers each feature's predictive capability; therefore, may miss a feature that only has better predictive power when combined with another feature(s). We recommend HFS in this research.

5. For the interaction term A×B (Figure 3(e)), 10 groups (levels) are presented – five performance metrics used in conjunction with two cross-validation methods. The rankers based on AUC and PRC performed better than the rankers based on OA and DGM for both cross-validation methods, CV and CVR. The rankers based on BAM outperformed the rankers based on all other performance metrics for the CV method. In contrast, rankers based on BAM performed worst compared to all other performance metrics for the CVR method.

6. For the interaction term A×C (Figure 3(f)), 25 groups (rankers) are presented. The rankers that use the NB, MLP and LR learners and based on the AUC, PRC and BAM performance metrics outperformed the other rankers. This conclusion is consistent with the results obtained from the main factors A and C. Subsequent to an ANOVA test and multiple comparisons analysis with respect to BGM, the conclusions drawn were very similar to those when AUC is the response variable.

Table 8. *t*-test

p-value ↘	AUC	BGM
NB	0.5690	0.9212
MLP	0.2645	0.4595
KNN	0.0341	0.0650
SVM	0.8533	0.5256
LR	0.8037	0.7132

We also compared the performances (with respect to AUC and BGM) of the defect prediction models built with the subsets of features to those built with the complete set of features – Table 7. A *t*-test was used for the comparisons at the $\alpha = 0.05$ significance level. For each learner, we have two groups of classification models: one group was created by using the smaller subsets of features (six attributes in this study as explained earlier), while the other group was created by using the original dataset with 42 software attributes. The *t*-test results are presented in Table 8. For all the learners, except KNN, the classification performances are very similar (p-values are less than 0.05) regardless of whether the complete set of attributes were used to build the prediction models or when the selected subsets of features were used to build the prediction models. In the case of KNN, the models constructed using the complete set of attributes performed better than the models constructed using the smaller subsets of features.

5 Related Work

We provide a brief overview of the most relevant works on feature selection, primarily in the software engineering field. The reader is referred to the cited references for further details on the respective works – due to the paper size.

Guyon and Elisseeff [3] present feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods. Liu and Yu [11] present an integrated approach to intelligent feature selection. Based on datasets from the UCI machine learning repository, Hall and Holmes [4] investigate six feature ranking techniques (in association with the C4.5 and Naïve Bayes learners) and conclude that wrappers were generally better in terms of accuracy, but not in terms of computational complexity. Some other works include: Saeys et al. [13], Jong et al. [7], and Ilczuk et al. [6].

The application of feature selection to problems in the software quality and reliability engineering is rather limited. Chen et al. [1] study wrapper-based techniques for software cost estimation, concluding that the data reduction improved estimations. Rodríguez et al. [12] evaluate three filter- and three wrapper-based models for software metrics and defect data sets, with the conclusion that wrappers were better than filters but only at a high computational cost. Compared to our work, Rodríguez et al. [12] base their conclusions on cross-validation training performance alone. We use three independent test data sets for model evaluation, in addition to using cross-validation for model training. Using independent test data sets during feature selection for defect prediction is relatively unique to our study.

6 Conclusion

A system with a long operational life is associated with practical issues related to software defect prediction, such as likely variations in the development process, data collection strategy, software quality goals, etc. from one development cycle to another. The software attributes and defect data collected from each source (a given release) provide unique perspectives into quality characteristics of that development project. Thus, mining project data from multiple system releases becomes vital to building a useful defect predictor.

This paper investigates feature selection for software quality modeling. A large case study of software metrics and defect data compares the attribute selection performances of wrapper-based feature ranking techniques (WRT) and our proposed hybrid feature selection (HFS) technique. Four large software measurement data sets obtained from a real-world system are used in the study. Our results reveals that among the two attribute selection techniques, HFS outperforms WRT. A comparison of the two cross-validation strategies indicates that cross-validation by itself is better than cross-validation with risk impact consideration. For the case study, even the removal of over 85% of software metrics did not have a negative effect on the consequent model's performance.

Further work will include empirical analysis with additional feature selection techniques in the context of data from other software development projects. A comparison of wrapper-based ranking and filter-based ranking would also benefit software quality practitioners.

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse. An empirical study of learning from imbalanced data using random forest. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 310–317, Washington, DC, USA, 2007. IEEE Computer Society.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Y. Saeys, T. Abeel, and Y. V. de Peer. Robust feature selection using ensemble feature selection techniques. In *ECML/PKDD (2)*, pages 313–325, 2008.

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