A Comparative Study of Feature-Ranking and Feature-Subset Selection Techniques for Improved Fault Prediction

Santosh Singh Rathore
Indian Institute of Information Technology, Design and Manufacturing Jabalpur, INDIA
santosh.rathore@iiitdmj.ac.in

Atul Gupta
Indian Institute of Information Technology, Design and Manufacturing Jabalpur, INDIA
atul@iiitdmj.ac.in

ABSTRACT

The quality of a fault prediction model depends on the software metrics that are used to build the prediction model. Feature selection represents a process of selecting a subset of relevant features that may lead to build improved prediction models. Feature selection techniques can be broadly categorized into two subcategories: feature-ranking and feature-subset selection. In this paper, we present a comparative investigation of seven feature-ranking techniques and eight feature-subset selection techniques for improved fault prediction. The performance of these feature selection techniques is evaluated using two popular machine-learning classifiers: Naive Bayes and Random Forest, over fourteen software project's fault-datasets obtained from the PROMISE data repository. The performances were measured using F-measure and AUC values. Our results demonstrated that feature-ranking techniques produced better results compared to feature-subset selection techniques. Among, the feature-ranking techniques used in the study, InfoGain and PCA techniques provided the best performance over all the datasets, while for feature-subset selection techniques ClassifierSubsetEval and Logistic Regression produced better results against their peers.

Keywords

Software metrics, feature selection, fault prediction, feature-ranking, Wrappers, Filters.

1. INTRODUCTION

Software fault prediction is a technique to identify the fault-prone modules before the testing phase by using the underlying properties of the software system. It aims to predict fault prone software modules in order to streamline the efforts to be applied in the later phases of software development. Fault prediction models are generally constructed by identifying the relationship between the structural measures[1] (software metrics) of the software such as coupling, cohesion, complexity etc. with faults. These models quantitatively describe how the internal structural properties relate to relevant external software quality factors such as fault proneness. However, there are some critical issues those need to be resolved before using the fault prediction results to guide the quality assurance process. One important concern is about identifying a subset of software metrics that are significantly correlated with fault proneness.

Many investigations in software fault prediction have reported an unusually higher misclassification rate that might be attributed to some inherent issues associated with the project datasets [1] [2]. One of the issue may be that the dataset typically stuff with unnecessary information, and therefore, they need to be preprocessed before using them for fault prediction [3]. It may be the case that some of the metrics may be redundant or worse, have an confounding effect on the fault proneness results. Earlier studies in this regard have confirmed that a high number of features (attributes) may lead to lower classification accuracy and higher misclassification errors [4] [5]. Higher dimensional data can also be a concern for many classification algorithms due to its high computational cost and memory usage [6]. Since the quality of the fault prediction model is relying on the set of software metrics used for model building process, the selection of right set of software metrics, best representing the fault dataset is a vital and foremost important task.

Feature selection is a process of selecting a subset of relevant features such that the quality of the prediction model can be improved. Each feature selection technique assesses the available feature's space and derives a relevant set of features. The criteria used to search the useful features depend on the nature of the technique used. Feature-ranking techniques rank each individual feature according to some decisive factor and then the analyst selects some of the features that are appropriate for a given dataset. While, feature-subset selection techniques search for subset of features that collectively have good predictive capability. In this paper, we perform a comparative study of the feature-ranking and feature-subset selection techniques to investigate their effectiveness in selecting a relevant set of features. We performed our study using a search-based approach, where we examine a suite of object-oriented (OO) design level metrics by applying feature selection techniques one by one and search for an optimal subset of metrics.

ISEC'14 February 19 - 21 2014, Chennai, India
Copyright 2014 ACM 978-1-4503-2776-3/14/02...$15.00.
http://dx.doi.org/10.1145/2590748.2590755.

[1]In this paper, we use terms 'Feature', 'Attribute' and 'Metric' interchangeably. All of refer a software metric.

Our comparative study includes seven different feature-ranking techniques: Chi-square (CS), Info Gain (IG), Gain Ratio (GR), ReliefF (RF), SVM, OneR and Principal Component Analysis (PCA), and eight different feature-subset selection techniques: two form of ClassifierSubsetEval (Rank search and Race search based), Wrapper, two form of FilterSubsetEval (Best search and Genetic search based), two form of CFSSubsetEval (Best search and Genetic search based) and one is of Logistic Regression analysis [7] [2] [8]. In order to evaluate the effectiveness of these techniques, we built prediction models using two different machine learning classifiers on reduced subsets of software metrics over the fourteen software project's fault-datasets, obtained from PROMISE data repository [9]. The two classifiers used in the study are: Naive Bayes (NB) and Random Forest (RF). Each prediction model is assessed with two performance measures: F-measure and the area under the Receiver Operating Characteristic (ROC) curve (AUC).

Our results demonstrated that feature-ranking techniques performed better compared to feature-subset selection techniques. Among, the used feature-ranking techniques, InfoGain (for NB) and PCA (for RF) techniques performed the best on average for all datasets, while for feature-subset selection techniques ClassifierSubsetEval (for RF) and Logistic Regression (for NB) produced the best results on average.

The rest of the paper is organized as follows. Section 2 presents the objective and the research questions for the study. Section 3 presents the details of various elements including the feature selection techniques, classifiers used for the investigation, the performance measures used in the study, information about the software fault datasets and software metrics used. Section 4 presents our experimental procedure. Section 5 has the results of our investigation. Section 6 is the discussion section. Related work is given in Section 7 and last section has a summary of the work.

2. RESEARCH QUESTIONS

The objective of this work is to perform a comparative investigation of seven feature-ranking techniques and eight feature-subset selection techniques to evaluate their effectiveness for feature selection. In this regards, we frame our research questions as follows:

RQ1. *Which feature-ranking technique works best overall?*

The performance of a feature-ranking technique depends on the nature of the fault dataset. Each technique uses different criteria to rank the available features. To that, we compared each of the seven feature-ranking techniques using AUC and F-measure values and find out the best of them.

RQ2. *Which feature-subset selection technique works best overall?*

Similarly, we compared each of the eight feature-subset selection techniques using AUC and F-measure values, and find out the best of them.

RQ3. *How does the feature-subset selection techniques compare with feature-ranking techniques?*

Here, we would like to evaluate the feature-subset selection and feature ranking techniques and investigate whether one of these approaches is better or if they both perform equally well.

RQ4. *How do classifiers affect the feature selection techniques?*

It may be possible that some feature selection techniques work very well with specific classifier, while worsen their performance with other classifiers. Thus, we evaluate the feature selection techniques using two different classifiers: Naive Bayes and Random Forest, and check their performance across the classifiers.

RQ5. *On an average, which feature selection technique work more effective than others?*

Some time a feature selection technique produce great results a specific set of datasets, but it does not work well on average for all the datasets. This question, investigate the average performance of each feature selection technique and find out the best of them for on average good performance.

3. ELEMENTS OF EXPERIMENTAL STUDY

Feature selection is a process of selecting a subset of relevant features for building robust classification models [6]. A feature selection algorithm is the combination of a search technique for proposing new feature subsets, along with an evaluation measure which scores the different feature subsets [2]. Wrappers refer to algorithms that use feedback from a learning algorithm to determine which attribute(s) to be use in building a classification model [2], whereas in filters, the attributes are selected using a method that is independent of an induction algorithm to determine which attributes are most relevant [2]. In broad way, feature selection can be categorized into two groups: feature-ranking techniques and feature-subset selection techniques.

3.1 Feature-Ranking Techniques

Feature-ranking techniques rank features independently without involving any learning algorithm. Feature ranking consists of scoring each feature according to a particular method, then selecting features based on their scores [10]. In this study, we use seven feature-ranking techniques. They describe below:

1. **ChiSquaredAttributeEval**: Assesses the importance of an attribute by calculating the value of the chi-squared statistic with respect to the class [11].

2. **GainRatioAttributeEval**: Assesses the value of an attribute by measuring the gain ratio with respect to the class [12].

3. **InfoGainAttributeEval**: Assesses the importance of an attribute by measuring the information gain with respect to the class [12].

4. **RelifeFAttributeEvel**: Assesses the value of an attribute by recurrently sampling an instance, and taking into consideration the value of the given attribute for the nearest instance of the same and different class [13].

5. **SVMAttributeEval**: An SVM classifier is used to assessed the value of an attribute. Attributes are ranked

by the square of the weight assigned by the SVM (support vector machine). For multiclass problems, attribute selection is performed by ranking attributes for each class independently using a one-vs-all method, and then processing the top of each pile to determine a final ranking [14].

6. **OneRAttributeEval**: Assesses the value of an attribute by using the OneR classifier.

7. **Principle Component Analysis (PCA)**: Principal component analysis is a technique to identify the underlying, orthogonal dimensions that explain relations between the variables in the dataset [15]. Each PC is a linear combinations of the standardized variables, i.e., software metrics. The first PC is the linear combination of all standardized variables that explain a maximum amount of variance in the data set. The second and subsequent PCs are linear combinations of all standardized variables. To better identify significant variables, the loadings of the variables in a given PC can be considered. The loading of a variable is its correlation with the PC. In order to further ease interpretation of the PCs, we can consider the varimax rotation also.

3.2 Feature-Subset Selection Techniques

Feature-subset selection techniques generate subset of attributes that collectively have good predictive capability. They are based on the assumption that a given attribute may have better predictive power when combined with some other attributes, compared to when used by itself. In this study, we use eight feature-subset selection techniques. They given below:

1. **ClassifierSubsetEval**: ClassifierSubsetEval assess a subset of attributes by the level of consistency in the class values when the training instances are projected onto the subset of attributes [16]. Commonly it used a search technique, which generates a small subset of attributes that will assess using the evaluation technique.

2. **CfsSubsetEval**: A subset of features is evaluated by taking into consideration the individual predictive capability of each feature along with the degree of redundancy between them. Subsets of features which are highly correlated with the class while having low inter-correlation are preferred [17].

3. **FilteredSubsetEval**: A method for running a random subset evaluator on data that has been passed through an arbitrary filter. The structure of the filter is entirely based on the training data [18].

4. **WrapperSubsetEval**: Evaluates attribute sets by using a learning algorithms. Cross validation is used to estimate the accuracy of the learning scheme for a set of attributes [18].

5. **Logistic Regression**: Logistic Regression (LR) can be used for the feature selection. Contrary to the conventional feature selection techniques (Filters and Wrappers), it requires fewer assumptions. It may be preferred when the data distribution is not normal, or the group sizes are unequal [19]. We use LR along with the Spearman's correlation analysis for metrics subset selection. We assist the fault proneness of each metric separately by performing Univariate Logistic Regression (ULR) analysis and subsequently perform the pair-wise correlation among the metrics using Spearman's correlation analysis. Later, we construct Multivariate Linear Regression (MLR) models to further reduce the metrics and identify a group of metrics that are collectively more significant for fault proneness [20].

Typically, Feature-subset selection techniques make use of various search methods. Search methods traverse the attribute space to find a good subset. The performance of the feature-subset selection technique may vary with the used search technique. Therefore, we have used different searching methods as mentioned in Table 1. BestSearch performs greedy hill climbing with backtracking. It can search forward from the empty set of attributes, backward from the full set, or start at an intermediate point and search in both directions by considering all possible single-attribute additions and deletions [18]. RaceSearch, calculates the cross-validation error of competing attribute subsets. It has four different searches, forward selection, backward elimination, schemata search, and rank racing. GeneticSearch uses a simple genetic algorithm to search the attribute space. RankSearch sorts attributes using a single-attribute evaluator and then ranks promising subsets using an attribute subset evaluator [18]. A list of all applied combinations is given in Table 1.

Table 1: Search Methods for feature-subset seletion

Subset Evaluator	**Search Method**
ClassifierSubsetEval	RankSearch
ClassifierSubsetEval	RaceSearch
FilterSubsetEval	BestSearch
FilterSubsetEval	GeneticSearch
CFSSubsetEval	BestSearch
CFSSubsetEval	GeneticSearch
WrapperSubsetEval	RankSearch

3.3 Classifiers

In this study, the effectiveness of the feature selection techniques is evaluated by two machine learning classifiers: Naive Bayes (NB) and Random Forest (RF) [21]. The main objective of this study is to evaluate the effectiveness of the various techniques, which should be independent of the classifiers used. For this reason, the choice of classifiers is orthogonal with respect to the intended contribution. One more reason to use two classifiers is that they represent quite different approaches of learning and are relatively fast, state-of-the-art algorithms that are often used in fault prediction applications.

Naive Bayes algorithm is a simplified version of Bayes formula to decide which class a test instance belongs to [22].

It calculate the posterior probability of each class, given the feature values present in the instance; the instance is assigned to the class with the highest probability. Learning a naive Bayes classifier is straightforward and involves simply estimating the probability of attribute values within each class from the training instances.

Random Forest is an ensemble learning method for classification that construct a multitude of decision trees from training sample and outputting the class that is the mode of the classes output by individual trees [23]. It generates an internal unbiased estimate of the generalization error as the forest building progresses. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing. Learning from a Random Forest is a fundamentally different process than learning a Naive Bayes model. We have used the WEKA implementation of all the algorithms with default parameters.

3.4 Performance Evaluation Parameters

We have used two evaluation measures: F-measure and Area under ROC curve (AUC)[2] to evaluate the performance of each feature selection technique [24]. These performance measures are commonly used in the field of data mining to evaluate the effectiveness of prediction models. Each feature selection technique uses different criteria to select a relevant set of features. Some may try to have an equal misclassification rate, some may want to achieve a higher true positive rate (TPR), and others may merely want to maximize true positive samples. These performance measures provides the trade-off between true positive rate and false negative rate.
F-measure: It is measured as the harmonic mean of precision and recall.

$$F-measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{1}$$

ROC curve: An ROC curve provides visualization of the tradeoff between the ability to correctly predict fault-prone modules (PD) and the number of incorrectly predicted fault free modules (PF).

3.5 Software Project Datasets

The datasets used in our study have been collected from the PROMISE data repository [9]. These datasets contained object-oriented (OO) metrics and faults found in the software modules during testing and after their release. The fault data was collected during requirements, design, development, unit testing, integration testing, system testing, beta release, controlled release, and general release of each release of the software system and was recorded in a database associated with the software. We have used four projects namely Camel, Ivy, Xerces and Poi with their fourteen successive releases. All the used software projects are the apache product[3] and have been implemented in the *Java programming language.* The reason to choose these software project is that they all followed the same structural properties and developed under the same environment. Each dataset contain the information of twenty OO metrics available at class level along with the fault information (number of faults). A detailed description of the datasets is tabulated in Table 2.

3.6 Software Metrics

To perform our experimental investigation, we have used the eighteen measures of coupling, cohesion, inheritance, encapsulation and complexity of an object-oriented software system. The metrics that are used for study are as follow- WMC, CBO, RFC, DIT, NOC, IC, CA, CE, MFA, LCOM, LCOM3, CAM, MOA, NPM, DAM, AMC, LOC and maxCC. We dropped CBM and AvgCC due to redundancy issues. Since, we have performed our study in the context of object-oriented software development and these metrics covered a broad range of object-oriented design features. These metrics are available in PROMISE data repository that encourage us to include them in our study. The detail description of these metrics is given in [25].

4. THE EXPERIMENTAL PROCEDURE

The whole procedure of comparative investigation for identifying the subset of software metrics is shown in figure 1.

Step 1: *The seven feature-ranking techniques are applied to the all the datasets. Each of the techniques will rank the attributes based on their performance. Further, we select $\lceil log2n \rceil$ of metrics out of n metrics.* We apply feature-ranking techniques over all fourteen project datasets to obtain the ranking of each attribute. First six feature-ranking techniques are filter-based techniques, implemented in WEKA machine learning tool[4]. We select the top $\lceil log2n \rceil$ attributes as the subsets of attributes, where n is the number of software metrics in the original dataset (in our study n= 18). An earlier study of Kehan Geo et al. [4] showed that it is appropriate to select first $\lceil log2n \rceil$ features of feature-ranking techniques when the WEKA implementation of these techniques is used. We also apply the PCA technique to on the same fourteen project datasets to select a subset of metrics. PCA technique generates number of the principal components (PCs) that are the linear combinations of the software metrics. We select only those PCs that have Eigenvalue more than 1. To further reduce the PCs, we use the loading factor value and discard the metrics those have loading factor less than 0.7.

Step 2: *The eight feature-subset selection techniques are applied to the same fourteen datasets. Each feature-subset selection technique is used with a classifier to select the most appropriate subset of attributes.* For feature-subset selection, we apply seven wrapper based feature-subset evaluator techniques and one Logistic Regression analysis based technique (Section 2). Each wrapper technique is used with some searching algorithm as described in Table 1. We use the WEKA implementation of these techniques to fit the training datasets. We also apply a Logistic Regression base technique for feature-subset selection [20]. Here, first we perform binary univariate logistic regression (ULR) analysis by considering fault-proneness as the dependent variable. To check the level of significance of each metric, we used three parameters of LR model: (i) Regression coefficient, (ii) Significance level (p- value); and (iii) Odds ratio. Next, we

[2]Wikipedia, http://en.wikipedia.org/wiki/Precision and recall

[3]Apache, http://www.apache.org/foundation/

[4]WEKA Machine Learning Tool, http://www.cs.waikato.ac.nz/ml/weka/

Table 2: Datasets used in the study

Project Name	No. of Modules	No. of non-commented LOC	No. of Faulty Modules	Total no. of Faults
Camel 1.0	340	33721	13	14
Camel 1.2	609	66302	216	522
Camel 1.4	873	98080	146	335
Camel 1.6	966	113055	187	500
Xerces 1.2	441	159254	70	115
Xerces 1.3	454	167095	68	193
Xerces 1.4	589	141180	435	1596
Ivy 1.1	111	27292	61	233
Ivy 1.5	241	59286	14	18
Ivy 2.0	352	87359	39	56
Poi 1.5	238	55428	141	342
Poi 2.0	315	93171	37	39
Poi 2.5	386	119731	248	496
Poi 3.0	442	129327	281	500

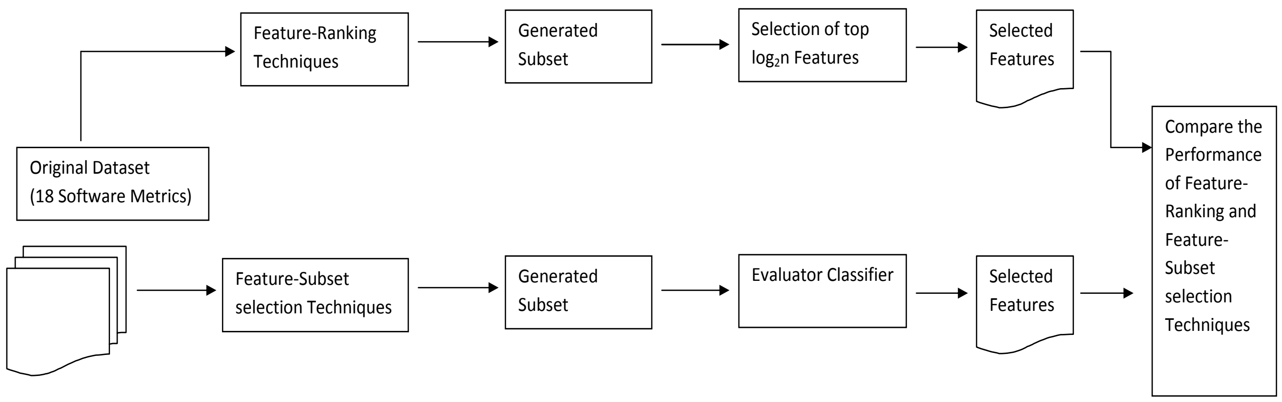
Feature-Ranking Techniques
Generated Subset
Selection of top log₂n Features
Selected Features
Original Dataset (18 Software Metrics)
Feature-Subset selection Techniques
Generated Subset
Evaluator Classifier
Selected Features
Compare the Performance of Feature-Ranking and Feature-Subset selection Techniques

Figure 1: The Overview of the Experimental Procedure

perform a pairwise Spearman's correlation analysis among the significant metrics and check for both positive as well as a negative correlation between metrics. If a metric shows higher correlation with other metrics, then we check the performance of these metrics individually and in the combine basis for fault prediction and select a metric or group of metrics, whosoever perform better. Finally, we perform multivariate linear regression analysis and select a best possible subset of metrics.

Step 3: *The obtained subsets of metrics from the above two steps are evaluated using two machine learning classifiers.* After performing the first two steps, we use two machine learning classifiers: NB and RF to build prediction models on the training datasets with various selected subsets of metrics. This step aims to evaluate the effectiveness of reduced subsets of metrics for fault prediction.

In this study, we perform all the experiments using 10 folds cross validation implemented in the WEKA machine learning tool. Each feature selection technique (both feature-ranking and feature-subset selection) was used with each of the fourteen datasets. Therefore, a total of 420 (15 techniques × 14 datasets × 2 classifiers) distinct prediction models were built in the study.

5. RESULTS

The experimental results are provided in this section. All the prediction models were build using the WEKA datamining tool and MS-Excel tool used for statistical comparison. Each of the feature selection techniques except Logistic Regression were implemented on WEKA. The Logistic Regression was implemented using IBM-SPSS tool.

5.1 Feature-Ranking Techniques

Procedurally, we have applied all seven feature-ranking techniques and eight feature-subset selection techniques over the datasets containing an original set of metrics one-by-one. Each of these techniques selects a subset of metrics based on the underlying formulations. Subsequently, we used the selected metrics subset to build the prediction models using

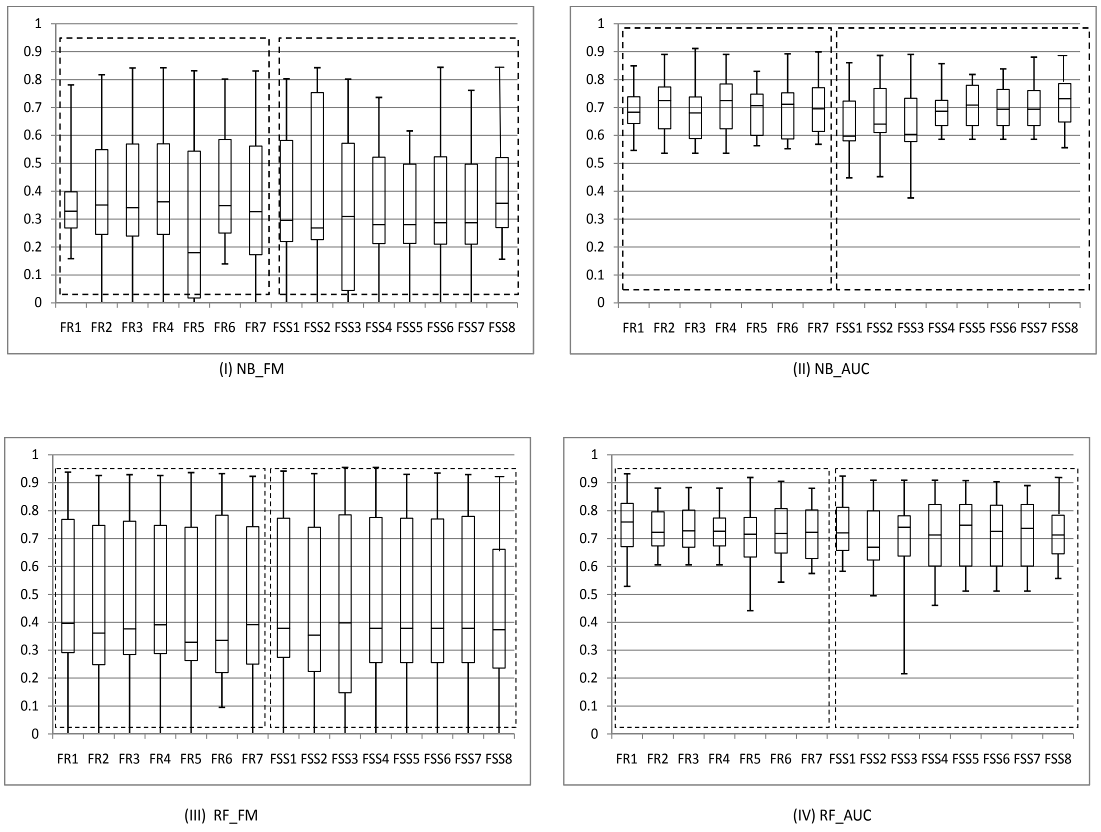

Figure 2: Box-Plot Analysis of the Feature Ranking and Feature Subset Selection Techniques

two machine-learning techniques. The performance of each prediction model in evaluated in terms of F-measure and AUC.

Table 3: Used Naming Conventions for Feature selection Techniques

Abbreviation	Corresponding Name
FR1	PCA
FR2	ChiSquare
FR3	GainRatio
FR4	InfoGain
FR5	RelifeE
FR6	SVM
FR7	OneR
FSS1	ClassifierSubsetEval (with Rank Search)
FSS2	ClassifierSubsetEval (with Race Search)
FSS3	WrapperSubsetEval (with Rank Search)
FSS4	FilterSubsetEval (with Best Search)
FSS5	FilterSubsetEval (with Genetic Search)
FSS6	CFSSubsetEval (with Best Search)
FSS7	CFSSubsetEval (with Genetic Search)
FSS8	Logistic Regression

Figure 2 shows the box-plot diagrams for each of the cases. The figure contained fifteen different box-plots, one for each feature selection technique. The Y-axis of the box-plot diagrams shows the name of the feature selection techniques used. For the sake of simplicity, we numbered each feature selection technique from FR1-FR7 (for feature-ranking), and FSS1-FSS8 (for feature-subset) left to right. Table 3 shows the mapping of these abbreviations to their actual names. Since, we have used two different classifiers and two performance measures. Therefore, we have four box plot diagrams (one for each combination). Each box-plot diagram is partitioned into two parts: One part for feature-ranking and one for feature-subset selection using the dotted rectangle. The left rectangle is for feature-ranking techniques and right for the feature-subset selection techniques. This simplification helps the reader to observe all the results by seeing only a single diagram. From the box-plot diagram, following trends are observed.

Among the used feature-ranking techniques, for Naive Bayes classifier, InfoGain techniques produced the best result for both the performance measures; F-measure and AUC. It has the highest max value compared to other techniques. For Random Forest classifier, PCA technique works best for both the performance measures. It has the highest median and max values compare to others.

5.2 Feature-Subset Selection

Similarly, among the used feature-subset selection techniques, for Naive Bayes classifier, Logistic Regression produced the best result for both the performance measures. It reported the highest median and max values compared to other techniques. For Random Forest classifier, ClassifierSubsetEval works best for both performance measures.

5.3 Overall Performance

To determine whether any of the feature selection approach is works better or they both are performed equally well, we employed a pairwise t-test:Paired Two Sample for Means, between the best feature-ranking technique and the best feature-subset selection technique. Since, we considered two different classifiers over fourteen software project datasets with two performance measures, a total number of four sets of pairwise comparisons (one for each classifier with performance measure) are used, each with 14 data points. The t-testing includes the best feature-ranking versus the best feature-subset selection techniques for F-measure and AUC. The results of t-test analysis are summarized in Table 4.

The first column of the table lists the performance measures, the second column shows the means difference values, the rest of the columns show the results of the statistical t-test. Our, t-test results show that there is not any significant difference between these two approaches, due to the p-value greater than .05. But, by looking the mean difference value, we found out that mean values of feature-ranking approach is better compared to feature-subset selection approach for all the performance measures (except one). Which, signify that performance of feature-ranking approach is better compared to feature-subset selection.

Table 5 has the results of t-test analysis, includes the on-average best performance of feature-ranking verses feature-subset selection techniques. Here, we first calculated the average performance value of each feature selection technique, and then used these values to perform t-test. Since, we have calculated average value over all fourteen datasets. Therefore, each feature selection gives one data point of average performance for each dataset and collectively we have 14 data points of all feature selection techniques corresponding to one classifier. We have used two classifiers with two performance measures, this gives us four sets, each with 14 data points to be compared.

From the table, it is clear that our t-test did not find any significant difference between feature-ranking techniques and feature-subset selection techniques for on-average performance. Again, we looked into the means difference values and found that feature-ranking techniques works well compared to feature-subset selection techniques for all four cases. This confirms the effectiveness of feature-ranking techniques for overall best performance.

5.4 Interaction of Classifiers with the Feature Selection Techniques

To determine the affect of classifiers over the performance of feature selection techniques, we have performed our investigation using two different classifiers, NB and RF. We found out that the selection of classifier affects the performance of the feature selection techniques. Our results show that for each used classifier, in cases of NB, InfoGain and Logistic Regression produced the best results, while for RF, PCA and ClassifierSubsetEval work best. This observation left an open question of investigation that *are feature selection techniques interact with the classifier algorithm?*, which can be answered in the future. Based on the finding of our experimental study, the answers to our research questions are summarized in Table 6.

6. DISCUSSION

In this paper, we were performing a comparative investigation of seven feature-ranking techniques and eight feature-subset selection techniques. The effectiveness of the techniques was evaluated using two machine learning classifiers over the fourteen software fault datasets. Among the used feature-ranking techniques, two techniques: InfoGain and PCA provided the best performances and therefore are recommended. For feature-subset selection techniques, two techniques: ClassifierSubsetEval and Logistic Regression performed better than other techniques and therefore are recommended. In summary, the experiments demonstrated that the feature-ranking techniques outperform feature-subset selection techniques.

In our study, we have used Logistic Regression (LR) as a feature-subset selection algorithm. We have applied the LR technique over the fourteen datasets and select a subset of metrics out of total metrics. The advantages of using LR is that contrary to the conventional feature selection techniques, it has fewer restrictive assumptions, and able to reduce the features substantially without any significant decrease in the classification accuracy [26] [27] [28]. Our experiment results show that Logistic Regression produced the better results compared to conventional feature selection techniques.

Threats to Validity:

Our models are built and evaluated on datasets available in public data repository. The system developed in the organization may pass the different effort pattern. Although, we believe that data available in the PROMISE data repository is correct and noise free. One needs to analyze the domain of the dataset, before using it for the experiment. The analysis results presented in this paper are based on the selection of certain classification algorithms and some performance measures. Although, we made our best effort to ensure that the selection of classification algorithms does not produce much influence on the conclusions. However, analysts should not rule out considering other classification algorithms when applying the proposed approach.

Our experimental study involving the use of the statistical analysis tools namely, WEKA, IBM-SPSS and the data collected from the publicly available software data repository. The fault densities and their distribution should depend on the fault data. Any biasing in this may influence the finding of our results. We have used various confusion matrix parameters to evaluate the performance of feature selection techniques and run all the experiments using WEKA tool. Here, we have used standard statistical data analysis, which includes a graphical method. One need to be understood the characteristics and the distribution of the dataset before applying the techniques for a project for the new domain.

7. RELATED WORK

An important issue associated with fault datasets in practice is the problem of having too many metrics (attributes). Simply put, not all metrics are likely to be necessary for accurate classification and include them in the prediction model may in fact lead to a worse model [4] [5]. There

Table 4: t-test: Best Feature-ranking Versus Best Feature-Subset Selection Techniques

Performance Measure	Mean Difference (Feature ranking - Feature subset selection)	t-value	t-critical	p-value
F-measure (NB)	0.02125202	0.319424077	1.770933383	0.377239966
AUC (NB)	-0.001428517	-0.12644443	1.770933383	0.450657434
F-measure (RF)	0.00009834	0.007042371	1.770933383	0.497243986
AUC (RF)	0.012214286	0.822620873	1.770933383	0.212777993

Table 5: t-test: Feature-ranking Versus Feature-Subset Selection Techniques (On-Average values)

Performance Measure	Mean Difference (Feature ranking - Feature subset selection)	t-value	t-critical	p-value
F-measure (NB)	0.006232143	0.296437757	1.770933383	0.385789319
AUC (NB)	0.01647992	1.214437514	1.770933383	0.123090321
F-measure (RF)	2.007248824	0.716673808	1.770933383	0.24312613
AUC (RF)	0.010669658	0.688920612	1.770933383	0.251491961

some work have been reported for solving the subset selection problem in order to identify the significant software metrics.

Guyon et al. [2] highlighted the key approaches used for attribute selection, including feature construction, feature ranking, multivariate feature selection, efficient search methods and feature validity assessment methods. They concluded that sophisticated wrapper or embedded methods improve predictive performance compared to simple variable ranking methods like correlation methods, but the improvements are not always significant: domains with large numbers of input variables suffer from the curse of dimensionality and multivariate methods may over fit the prediction model.

Harman et al. [29] provided a comprehensive survey of the studies related to search based software engineering. They identified research trends and relationships between the techniques applied, the applications to which they have been applied and highlighted gaps in the literature and avenues for further research.

Rodriguez et al. [5] performed an investigation using feature selection algorithms with three filter models and three wrapper models over four software project datasets. They concluded that the reduced datasets maintained the prediction capability with fewer attributes than the original datasets. In addition, while it was stated that the wrapper model was better than the filter model, it came at a high computational cost.

Liu and Yu [6] provided a survey of feature selection algorithms and presented an integrated approach of intelligent feature selection. Their study introduced concepts and algorithms of feature selection, survey existing feature selection algorithms for classification and clustering, groups and compares different algorithms with a categorizing framework based on search strategies. Their evaluation criteria, and data mining tasks, reveal unattempted combinations, and provides guidelines in selecting feature selection algorithms. They stated that as data mining develops and expands in new application areas, feature selection also faces new challenges that need to be further researched.

Khoshgoftaar et al. [4] reported a study of selecting software metrics for defect prediction. Their study focused on the problem of attribute selection in the context of software quality estimation. They presented a comparative investigation for evaluating their proposed hybrid attribute selection approach. Their results demonstrated that the automatic hybrid search algorithm performed the best among the feature subset selection methods. Moreover, performances of the defect prediction models either improved or remained unchanged when over 85% of the software metrics were eliminated

The limitation of the studies discussed above is that no systematic comparative study was performed to evaluate the effectiveness of different feature selection techniques. In addition, very few studies have investigated feature selection techniques in the context of software fault prediction. Since, the characteristics of dataset in some specific domains influence the feature selection techniques, and the pattern of attributes found in one domain may not appear in other domains like software metrics. Hence, it is needed to evaluate the performance of the feature selection techniques for software fault datasets also to understand how well these feature selection techniques handle and best manage the different challenges posed by it. In our study, we compare the performance of fifteen feature selection techniques in the context of software engineering. To validate the effectiveness of the feature selection techniques, we use two classifiers and evaluate the prediction models using two performance measures, F-measure and AUC.

8. CONCLUSIONS

Software fault prediction can be helpful to identify the fault-prone software modules to streamline the efforts applied in the later phases of software development. Since, the quality of constructed fault prediction model is highly influenced by the quality of the fault dataset, which consist software metrics and the faults information. Hence, the selection of the right set of metrics becomes an important step of the improved fault prediction process. In this study, we presented an extensive investigation of fifteen feature selection techniques. We categorized the feature selection tech-

Table 6: Summarized Results

Research Questions		Experimental Findings
RQ1	Which feature-ranking technique works best overall?	We found that InfoGain produced the best results for NB classifier over all the datasets. While PCA is best for RF classifier.
RQ2	Which feature-subset selection technique works best overall?	We found that Logistic Regression produced the best results for NB classifier over all the datasets. While ClassifierSubsetEval is best for RF classifier.
RQ3	How does the feature-subset selection techniques compare with feature-ranking techniques?	Our t-test did not found any significant difference between feature-ranking and feature-subset selection techniques. However, the mean difference value shows that feature-ranking outperform the feature-subset selection techniques for all the performance measures.
RQ4	How do classifiers affect the feature selection techniques?	We found that the performance of the feature selection techniques is varied with the different classifiers used. This shows that selection of classifier affect the feature selection techniques.
RQ5	On an average, which feature selection technique work more effective than others?	Our t-test over the average values of the AUC and F-measure shows that feature-ranking outperform the feature-subset selection techniques for all the performance measures.

niques into two groups, feature-ranking and feature-subset selection. Feature-ranking assesses each individual feature using some criterion and ranked them according to their performance. On the other hand, feature-subset selection techniques select subsets of metrics that collectively have good predictive capability.

Our study aimed to investigate, which feature-ranking and feature-subset selection technique produced the best results individually, and among the feature-ranking and feature-subset selection techniques, which one is working better. Our results showed that InfoGain and PCA collectively worked best for feature-ranking, and ClassifierSubsetEval and Logistic Regression collectively worked best for feature-subset selection. Also the feature-ranking techniques produced better results compared to feature-subset selection techniques. In addition, we have investigated that how does classifiers affect the feature selection techniques. We found that the performance of the feature selection techniques is varied with the classifiers used. This suggested for an interaction effect between feature selection techniques and classifiers. It will be useful to study this effect for a more informed use of a feature selection technique against a given classifier.

9. REFERENCES

[1] Kehan Gao, Taghi M. Khoshgoftaar, and Naeem Seliya. Predicting high-risk program modules by selecting the right software measurements. *Software Quality Control*, 20(1):3–42, March 2012.

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

[3] N. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, (5):675–689, 2000.

[4] G. Kehan, T. M. Khoshgoftaar, H. Wang, and N. Seliya. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software Practice and Experience*, 41(5):579–606, 2011.

[5] D. Rodriguez, R. Ruiz, J. Cuadrado-Gallego, J. Aguilar-Ruiz, and M. Garre. Attribute selection in software engineering datasets for detecting fault modules. In *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, EUROMICRO '07, pages 418–423, 2007.

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

[7] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, September 2007.

[8] R. Shatnawi and W. Li. The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process. *The Journal of Systems and Software*, (11):1868–1882, 2008.

[9] PROMISE Data Repository. `http://promisedata.org/`.

[10] María Tombilla-Sanromán Noelia Sánchez-Maroño, Amparo Alonso-Betanzos. Filter methods for feature selection Ű a comparative study. *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, 4881:178–187, 2007.

[11] R. L. Plackett. Karl pearson and the chi-squared test. *International Statistical Review*, 51(1):59–72, 1983.

[12] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[13] 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.

[14] NataŽa Milic-Frayling Janez Brank, Marko Grobelnik and Dunja Mladenic. Feature selection using linear support vector machines. In *TechReport, Microsoft Research*, pages 1–18, 2002.

[15] J. C. Munson and T. M. Khoshgoftaar. The dimensionality of program complexity. In *Proceedings of the 11th international conference on Software engineering*, ICSE '89, pages 245–253. ACM, 1989.

[16] Hiroshi Motoda Manoranjan Dash, Huan Liu. Consistency based feature selection. *Knowledge Discovery and Data Mining*, 1805:98–109, 2000.

[17] M. Indra Devi, R. Rajaram, and K. Selvakuberan. Generating best features for web page classification.

Webology, 5(1), 2008.

[18] Eibe Frank Ian Witten and Mark Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, third edition, 2011.

[19] Qi Cheng, P.K. Varshney, and M.K. Arora. Logistic regression for feature selection and soft classification of remote sensing data. *IEEE Letters on Geoscience and Remote Sensing*, 3(4):491–494, 2006.

[20] Santosh Singh Rathore and Atul Gupta. Validating the effectiveness of object-oriented metrics over multiple releases for predicting fault proneness. In *Proceeding of 19th Asia-Pecific Software Engineering Conference (APSEC 2012)*, pages 270–275, 2012.

[21] E. Arisholm, L. Briand, and E. B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *The Journal of Systems and Software*, (1):2–17, 2010.

[22] Harry Zhang. The Optimality of Naive Bayes. In Valerie Barr and Zdravko Markov, editors, *FLAIRS Conference*. AAAI Press, 2004.

[23] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[24] M ues C P ietsch S Lessmann S, Baesens B. Benchmarking classification models for s oftware defect prediction: A proposed framework and novel findings. *IEEE Transanction on Software Enggineering*, 34(4):485Ű496, 2008.

[25] M. Jureczko. Significance of different software metrics in defect prediction. *Software Engineering: An International Journal*, 1(1):86–95, 2011.

[26] Roman Zakharov and Pierre Dupont. Ensemble logistic regression for feature selection. In *Pattern Recognition in Bioinformatics*, volume 7036, pages 133–144. 2011.

[27] Mahesh Pal. Multinomial logistic regression-based feature selection for hyperspectral data. *International Journal of Applied Earth Observation and Geoinformation*, 14(1):214–220, 2012.

[28] K. Kampa, E. Hasanbelliu, and J. C. Principe. Closed-form cauchy-schwarz pdf divergence for mixture of gaussians. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 2578–2585, 2011.

[29] M. Harman, S. A. Mansouri, and Y. Zhang. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. In *Technical report: TR-09-03*. Department of Computer Science, KingŠs College London, UK, 2009.