2010 Ninth International Conference on Machine Learning and Applications

A Comparative Study of Ensemble Feature Selection Techniques for Software Defect Prediction

Huanjing Wang
huanjing.wang@wku.edu

Taghi M. Khoshgoftaar
taghi@cse.fau.edu

Amri Napolitano
amrifau@gmail.com

Abstract

Feature selection has become the essential step in many data mining applications. Using a single feature subset selection method may generate local optima. Ensembles of feature selection methods attempt to combine multiple feature selection methods instead of using a single one. We present a comprehensive empirical study examining 17 different ensembles of feature ranking techniques (rankers) including six commonly-used feature ranking techniques, the signal-to-noise filter technique, and 11 threshold-based feature ranking techniques. This study utilized 16 real-world software measurement data sets of different sizes and built 13,600 classification models. Experimental results indicate that ensembles of very few rankers are very effective and even better than ensembles of many or all rankers.

Keywords*: ensembles, feature ranking, defect prediction.*

1 Introduction

Classification models are commonly used to detect faulty software modules based on software metrics. Such models are trained using data collected prior to software testing and operations. An ideal classification model would be able to identify every fault-prone (fp) module correctly. In real-world problems, a learning algorithm is rarely capable of generating such a perfect model. Therefore, researchers and practitioners put forth much effort to generate the best possible model by maximizing the accuracy of its predictions. Previous studies have successfully shown that the performance of these models improves when irrelevant and redundant features are eliminated from the original data set [5, 15].

This study examined ensembles of feature selection methods for predicting faulty program modules. Using 18 different filter-based ranking techniques (rankers), we examined the performance of models with selected features using 17 different ensembles of rankers. Classification models were built using a well known classifier, Naïve Bayes, for software defect prediction [12]. To our knowledge, no previous comprehensive empirical investigation has been performed comparing the performance of 17 ensembles of rankers in the domain of software reliability engineering and perhaps other application domains. Our results show that ensembles of very few rankers usually perform similarly or even better than ensembles of many or all rankers. Finally, analysis of variance (ANOVA) was performed to evaluate the statistical significance of the results presented in the paper.

The remainder of the paper is organized as follows. Section 2 provides an overview of related work, while Section 3 presents 18 rankers and ensembles of feature selection techniques. Section 4 describes the data sets, experimental design, and experimental results. Finally, we conclude the paper in Section 5 and provide suggestions for future work.

2 Related Work

Feature selection has been applied in many data mining and machine learning applications. The main goal of feature selection is to select a subset of features that minimizes the prediction errors of classifiers. Guyon and Elisseeff [6] outlined 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 [7] investigated six attribute selection techniques that produce ranked lists of attributes and applied them to several data sets from the UCI machine learning repository. The application of feature selection to problems in the software quality and reliability engineering domain is rather limited. Rodríguez et al. [15] evaluated three filter- and three wrapper-based models for software metrics and defect data sets, with the conclusion that wrappers were better than filters but at a high computational cost.

Ensemble feature selection typically combines multiple models to form a single one. One of the main problems which needs to be considered when building an ensemble model is diversity. Diversity may be achieved through using different data sets, feature subsets, or classifiers. A recent study on ensembles of multiple feature ranking tech-

978-0-7695-4300-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICMLA.2010.27

niques was done by Olsson and Oard [13]. Their work considered combinations of three commonly used filter-based feature ranking techniques, including document frequency thresholding, information gain, and the chi-square method (χ^2_{max} and χ^2_{avg}) for text classification problems. Wang et al. [16] studied the ensembles of six commonly used filter-based rankers. In this paper, we combined rankers selected from 18 filter-based rankers and constructed 17 ensembles (with different number of rankers).

3 Ensemble Techniques

3.1 Filter-based Feature Ranking Techniques

Filter-based feature ranking techniques (rankers) rank features independently without involving any learning algorithm, then the best features are selected from the ranking list. Researchers have developed a large number of methods to rank features. In this study, we use 18 filter-based feature ranking methods, including six commonly-used feature selection methods, the signal to noise filter (S2N), and 11 threshold-based feature selection techniques.

The six well-known commonly-used filter-based feature ranking methods include: chi-square, information gain, gain ratio, two types of ReliefF, and symmetrical uncertainty. The chi-square - χ^2 (CS) test [14] is used to examine the distribution of the class as it relates to the values of the feature in question. The null hypothesis is that there is no correlation; each value is as likely to have instances in any one class as any other class. Information gain (IG), gain ratio (GR), and symmetrical uncertainty (SU) are measures based on the concept of entropy from information theory [17]. Relief is an instance-based feature ranking technique introduced by Kira and Rendell [10]. ReliefF is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the WEKA tool [17]. When the `WeightByDistance` (weight nearest neighbors by their distance) parameter is set as default (false), the algorithm is referred to as RF; when the parameter is set to 'true', the algorithm is referred to as RFW.

Signal-to-noise is a measure used in electrical engineering to quantify how much a signal has been corrupted by noise. It is defined as the ratio of signal's power to the noise's power corrupting the signal. The signal-to-noise (S2N) can also be used as feature ranking method [18]. For a binary class problem (such as fp, nfp), the S2N is defined as the ratio of the difference of class means ($\mu_{fp} - \mu_{nfp}$) to the sum of standard deviation of each class ($\sigma_{fp} + \sigma_{nfp}$). If one attribute's expression in one class is quite different from its expression in the other, and there is little variation within the two classes, then the attribute is predictive. Therefore S2N favors attributes where the range of the expression vector is large, but where most of that variation is due to the class distribution. S2N is rarely used as a feature ranking technique.

Algorithm 1: Threshold-based Feature Selection Algorithm

input :
1. Data set D with features $F^j, j = 1, \ldots, m$;
2. Each instance $x \in D$ is assigned to one of two classes $c(x) \in \{fp, nfp\}$;
3. The value of attribute F^j for instance x is denoted $F^j(x)$;
4. Metric $\omega \in$ {FM, OR, PO, PR, GI, MI, KS, DV, GM, AUC, PRC};
5. A predefined threshold: number (or percentage) of the features to be selected.

output:
Selected feature subsets.

for $F^j, j = 1, \ldots, m$ **do**
  Normalize $F^j \mapsto \hat{F}^j = \frac{F^j - \min(F^j)}{\max(F^j) - \min(F^j)}$;
  Calculate metric ω using attribute $\hat{F}^j$ and class attribute at various decision threshold in the distribution of $\hat{F}^j$. The optimal ω is used, $\omega(\hat{F}^j)$.

Create feature ranking $\mathbb{R}$ using $\omega(\hat{F}^j)\ \forall j$.
Select features according to feature ranking $\mathbb{R}$ and a predefined threshold.

Eleven threshold-based feature selection techniques (TBFS) were recently proposed and implemented by our research group within WEKA [17]. The procedure is shown in Algorithm 1. First each attribute's values are normalized between 0 and 1 by mapping F^j to $\hat{F}^j$. The normalized values are treated as posterior probabilities. Each independent attribute is then paired individually with the class attribute and the reduced two attribute data set is evaluated using 11 different performance metrics based on a set of posterior probabilities. In standard binary classification, the predicted class is assigned using the default decision threshold of 0.5. The default decision threshold is often not optimal, especially when the class is imbalanced. Therefore, we propose the use of performance metrics that can be calculated at various points in the distribution of $\hat{F}^j$. At each threshold position, we classify values above the threshold as positive, and below as negative. Then we go in the opposite direction, and consider values above as negative, and below as positive. Whatever direction produces the more optimal attribute values is used.

The true positive (TPR), true negative (TNR), false positive (FPR), false negative (FNR) rates can be calculated at each threshold $t \in [0, 1]$ relative to the normalized attribute $\hat{F}^j$. The threshold-based attribute ranking techniques we propose utilize these rates as described below.

- *F-measure (FM)*: is a single value metric derived from the F-measure that originated from the field of information retrieval. The maximum F-measure is obtained when varying the decision threshold value between 0 and 1.

- *Odds Ratio (OR)*: is the ratio of the product of correct (TPR times TNR) to incorrect (FPR times FNR) predictions. The maximum value is taken when varying the decision threshold value between 0 and 1.

- *Power (PO)*: is a measure that avoids common false positive cases while giving stronger preference for positive cases [4]. Power is defined as:

$$\text{PO} = \max_{t \in [0,1]} \left((TNR(t))^k - (FNR(t))^k \right)$$

where $k = 5$.

- *Probability Ratio (PR)*: is the sample estimate probability of the feature given the positive class divided by the sample estimate probability of the feature given the negative class [4]. PR is the maximum value of the ratio when varying the decision threshold value between 0 and 1.

- *Gini Index (GI)*: measures the impurity of a data set. GI for the attribute is then the minimum Gini index at all decision thresholds $t \in$[0, 1].

- *Mutual Information (MI)*: measures the mutual dependence of the two random variables. High mutual information indicates a large reduction in uncertainty, and zero mutual information between two random variables means the variables are independent.

- *Kolmogorov-Smirnov (KS)*: utilizes the Kolmogorov-Smirnov statistic to measure the maximum difference between the empirical distribution functions of the attribute values of instances in each class. It is effectively the maximum difference between the curves generated by the true positive and false positive rates as the decision threshold changes between 0 and 1.

- *Deviance (DV)*: is the residual sum of squares based on a threshold t. That is, it measures the sum of the squared errors from the mean class given a partitioning of the space based on the threshold t and then the minimum value is chosen.

- *Geometric Mean (GM)*: is a single-value performance measure which is calculated by finding the maximum geometric mean of TPR and TNR as the decision threshold is varied between 0 and 1.

- *Area Under ROC (Receiver Operating Characteristic) Curve* (AUC): has been widely used to measure classification model performance [3]. The ROC curve is used to characterize the trade-off between true positive rate and false positive rate. In this study, ROC curves are generated by varying the decision threshold t used to transform the normalized attribute values into a predicted class.

- *Area Under the Precision-Recall Curve* (PRC): is a single-value measure that originated from the area of information retrieval. The area under the PRC ranges from 0 to 1. The PRC diagram depicts the trade off between recall and precision.

Algorithm 2: Ensemble Feature Selection Algorithm

input :
1. Data set D with features F^j, $j = 1, \ldots, m$;
2. Each instance $x \in D$ is assigned to one of two classes $c(x) \in \{fp, nfp\}$;
3. k: number of selected rankers;
4. k selected rankers $\omega \in$ {CS, IG, GR, RF, RFW, SU, S2N, FM, OR, PO, PR, GI, MI, KS, DV, GM, AUC, PRC};
5. A predefined threshold: number (or percentage) of the features to be selected.

output:
Selected feature subsets.

for $i = 1, \ldots, k$ **do**
  Employ ω_i to rank features F^j, and get new rankings $\omega_i(F^j), j = 1, \ldots, m$.

Create feature ranking $\mathbb{R}$ by combining the k different rankings $\{\omega_i(F^j), i = 1, \ldots, k\}$ $\forall j$ with arithmetic mean.
Select features according to feature ranking $\mathbb{R}$ and a predefined threshold.

3.2 Ensemble Techniques

We investigated the use of ensembles of feature ranking techniques (rankers). These ranking techniques are combined to yield more stable and robust results. The procedure of the ensemble technique is presented in Algorithm 2. There are two essential steps in creating a single feature ranking list from multiple ranking lists. First, a set of different ranking lists is created using corresponding filter-based rankers and input to the next combining step; second, these ranking lists are integrated using rank ordering of the features. Diversity can be achieved by using various rankers. The combining methods used in the study is arithmetic mean, where each feature's score is determined by the average of the ranking scores of the feature in each ranking list. Finally, the highest ranked attributes are selected from the original data to form the training data set.

4 Experiments

4.1 Data Sets

The software metrics and fault data for this case study were collected from real-world software projects, including a very large telecommunications software system (denoted as LLTS) [5], the Eclipse project [19], and NASA software project KC1 [11].

LLTS contains data from four consecutive releases, which are labeled as SP1, SP2, SP3 and SP4. The software measurement data sets consist of 42 software metrics, including 24 product metrics, 14 process metrics, and four execution metrics [5].

From the PROMISE data repository [19], we also obtained the Eclipse defect counts and complexity metrics data set. In particular, we use the metrics and defects data at the software package level. The original data for Eclipse packages consists of three releases denoted 2.0, 2.1, and

Table 1. Software Data Sets Characteristics

	Data	#Metrics	#Modules	%fp	%nfp
LLTS	SP1	42	3649	6.28%	93.72%
	SP2	42	3981	4.75%	95.25%
	SP3	42	3541	1.33%	98.67%
	SP4	42	3978	2.31%	97.69%
Eclipse	Eclipse2.0-10	209	377	6.1%	93.9%
	Eclipse2.0-5	209	377	13.79%	86.21%
	Eclipse2.0-3	209	377	26.79%	73.21%
	Eclipse2.1-5	209	434	7.83%	92.17%
	Eclipse2.1-4	209	434	11.52%	88.48%
	Eclipse2.1-2	209	434	28.8%	71.2%
	Eclipse3.0-10	209	661	6.2%	93.8%
	Eclipse3.0-5	209	661	14.83%	85.17%
	Eclipse3.0-3	209	661	23.75%	76.25%
KC1	KC1-20	63	145	6.90%	93.10%
	KC1-10	63	145	14.48%	85.52%
	KC1-5	63	145	24.83%	75.17%

Table 2. Ensembles of Multiple Rankers

	Rankers																		
Ensembles	CS	IG	GR	RF	RFW	SU	S2N	FM	OR	PO	PR	GI	MI	KS	DV	GM	AUC	PRC
EM-2IR		X		X														
EM-2AM													X				X	
EM-2IA		X															X	
EM-2IM		X											X					
EM-2RA				X													X	
EM-2RM				X									X					
EM-3IRA		X		X													X	
EM-3IAM		X											X				X	
EM-3IRM		X		X									X					
EM-3RAM				X									X				X	
EM-4		X		X									X				X	
EM-6C		X		X									X	X		X	X	
EM-6O	X	X	X	X	X	X												
EM-8		X		X									X	X	X	X	X	X
EM-10		X		X		X	X						X	X	X	X	X	X
EM-11								X	X	X	X	X	X	X	X	X	X	X
EM-18	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

3.0 respectively. We transform the original data by: (1) removing all nonnumeric attributes, including the package names, and (2) converting the post-release defects attribute to a binary class attribute: fault-prone (*fp*) and not fault-prone (*nfp*). Membership in each class is determined by a post-release defects threshold λ, which separates *fp* from *nfp* packages by classifying packages with λ or more post-release defects as fp and the remaining as nfp. In our study, we use $t \in \{10, 5, 3\}$ for release 2.0 and 3.0 while we use $t \in \{5, 4, 2\}$ for release 2.1. All nine derived data sets contain 209 attributes. Releases 2.0, 2.1, and 3.0 contain 377, 434 and 661 instances respectively.

The NASA project, KC1 [11], includes 145 instances containing 95 attributes each. After removing 32 Halstead derived measures, we have 63 attributes. We used three different thresholds to define defective instances, thereby obtaining three structures of the preprocessed KC1 data set. The thresholds are 20, 10 and 5, indicating instances with numbers of defects greater than or equal to 20, 10, or 5 belong to *fp* class. The three data sets are named KC1-20, KC1-10, and KC1-5.

Table 1 lists the characteristics of the 16 data sets utilized in this work, which exhibit different distributions of class skew (i.e, the percentage of *fp* module).

4.2 Experimental Design

In our experiments, we investigated 17 different ensembles of feature selection methods. These ensembles are constructed by combining different selected rankers and different numbers of rankers. Among 18 filter-based ranking techniques, our previous study showed that IG, RF, AUC-based and MI-based TBFS have better performance than others [1]. We considered all combinations of these four rankers and constructed ensembles of two rankers (EM-2IR, EM-2AM, EM-2IA, EM-IM, EM-2RA and EM-2RM), ensembles of three rankers (EM-3IRA, EM-3IAM, EM-3IRM and EM-3RAM) and an ensemble of four rankers (EM-4). We also construct an ensemble of six rankers (EM-6C), an ensemble of 8 rankers (EM-8) and an ensemble of 10 rankers (EM-10) by adding two rankers each time to EM-4 from the remaining rankers in order of their performances. In addition, the ensemble of six commonly-used rankers (EM-6O), ensemble of 11 threshold-based feature selection methods (EM-11) and ensemble of all 18 rankers (EM-18) are also considered. Table 2 describes these 17 different ensembles of rankers.

In order to evaluate the effectiveness of a given ensemble technique, we built classification models using the feature subset selected by that ensemble approach. Classifiers were built with a well-known classification algorithm, Naïve Bayes. Naïve Bayes (NB) utilizes Bayes's rule of conditional probability and is termed 'naïve' because it assumes conditional independence of the features. Prior research has shown that the Naïve Bayes classifier often performs well, even on real-world data where the attributes are related [17].

The classification models are evaluated using the area under the ROC curve (AUC) performance metric. AUC is widely used, providing a general idea of the predictive potential of the classifier. A higher AUC is better, as it indicates that the classifier, across the entire possible range of decision threshold, has a higher true positive rate. It has been shown that AUC has lower variance and is more reliable than other performance metrics (such as precision, recall, F-measure) [8]. Note that AUC is used both to select the most predictive subset of features using TBFS and also to evaluate the classifiers constructed using this set of features.

During the experiments, ten runs of five-fold cross-validation were performed. For each of the five folds, one fold is used as test data while the other four are used as the training data. First, we ranked the attributes using the 17 ensembles of feature selection methods separately. Once the attributes are ranked, the top $\lceil \log_2 n \rceil$ (n is the number of independent features for a given data set) attributes are selected (as well as the class attribute) to yield final training data. The reasons why we select the top $\lceil \log_2 n \rceil$ features include (1) no general guidance has been found in related literature on the number of features that should be selected

Table 3. Classification Performance

	EM-2IR	EM-2AM	EM-2IA	EM-2IM	EM-2RA	EM-2RM	EM-3IRA	EM-3IAM	EM3-IRM	EM-3RAM	EM-4	EM-6C	EM-6O	EM-8	EM-10	EM-11	EM-18
SP1	0.7835	0.7727	0.7736	0.7819	0.7813	**0.7963**	0.7851	0.7780	0.7804	0.7813	0.7843	0.7758	0.7819	0.7799	0.7857	0.7762	0.7850
SP2	**0.8157**	0.8081	0.8066	0.8074	0.8149	0.8140	0.8106	0.8070	0.8107	0.8097	0.8078	0.8074	0.8109	0.8068	0.8051	0.8082	0.8084
SP3	**0.8273**	0.7984	0.7980	0.8106	0.8100	0.8261	0.8136	0.8053	0.8229	0.8136	0.8124	0.8114	0.8183	0.8086	0.8099	0.8095	0.8167
SP4	**0.8055**	0.7769	0.7832	0.7870	0.7982	0.7997	0.7982	0.7865	0.8020	0.7923	0.7980	0.7873	0.8039	0.7951	0.7970	0.7812	0.8027
Eclipse2.0-10	0.8617	0.8606	0.8622	0.8439	0.8508	**0.8690**	0.8483	0.8530	0.8599	0.8583	0.8562	0.8512	0.8670	0.8514	0.8472	0.8334	0.8560
Eclipse2.0-5	0.8838	0.8609	0.8591	0.8586	**0.8849**	0.8813	0.8740	0.8618	0.8750	0.8773	0.8663	0.8552	0.8760	0.8541	0.8564	0.8528	0.8564
Eclipse2.0-3	0.8022	0.7887	0.7956	0.7934	0.7999	0.8017	**0.8040**	0.7924	0.7986	0.7984	0.7968	0.7957	0.7999	0.7932	0.7946	0.7921	0.7948
Eclipse2.1-5	0.8585	0.8531	0.8563	0.8549	0.8641	0.8562	0.8660	0.8534	0.8652	0.8723	0.8506	0.8394	**0.8742**	0.8475	0.8441	0.8467	0.8500
Eclipse2.1-4	0.8420	0.8181	0.8192	0.8245	0.8545	**0.8581**	0.8478	0.8222	0.8431	0.8491	0.8345	0.8299	0.8508	0.8273	0.8271	0.8159	0.8283
Eclipse2.1-2	0.7971	0.7550	0.7551	0.7542	**0.8200**	0.7892	0.8174	0.7533	0.8023	0.8021	0.7877	0.7711	0.7972	0.7699	0.7698	0.7570	0.7668
Eclipse3.0-10	0.9139	0.8919	0.8992	0.8894	**0.9159**	0.9137	0.9145	0.8942	0.9098	0.9146	0.9041	0.9028	0.9123	0.9026	0.9002	0.8649	0.8956
Eclipse3.0-5	0.9068	0.8868	0.8851	0.8858	**0.9083**	0.9023	0.8983	0.8865	0.8985	0.8973	0.8911	0.8901	0.8950	0.8865	0.8848	0.8860	0.8851
Eclipse3.0-3	**0.8764**	0.8103	0.8104	0.8121	0.8747	0.8659	0.8434	0.8111	0.8421	0.8434	0.8265	0.8209	0.8441	0.8158	0.8188	0.8099	0.8126
KC1-5	0.7849	0.7633	0.7722	0.7557	**0.7907**	0.7782	0.7845	0.7541	0.7804	0.7794	0.7777	0.7792	0.7841	0.7732	0.7753	0.7393	0.7680
KC1-10	0.7605	0.7141	0.7237	0.7520	0.7551	0.7620	0.7606	0.7279	0.7674	0.7611	0.7590	0.7385	**0.7676**	0.7430	0.7477	0.7250	0.7490
KC1-20	0.8412	0.8479	0.8487	0.8523	0.8244	0.8444	0.8360	0.8459	**0.8574**	0.8379	0.8439	0.8496	0.8507	0.8486	0.8446	0.8353	0.8490
Average	**0.8351**	0.8129	0.8155	0.8165	0.8342	0.8349	0.8314	0.8145	0.8322	0.8305	0.8248	0.8191	0.8334	0.8190	0.8193	0.8083	0.8203

when using a feature ranking technique; (2) a software engineering expert with more than 20 years experience recommended selecting $\lceil \log_2 n \rceil$ number of metrics for software quality prediction; (3) a preliminary study showed that $\lceil \log_2 n \rceil$ was appropriate for various learners; and (4) a recent study [9] showed that it was appropriate to use $\lceil \log_2 n \rceil$ as the number of features when using WEKA [17] to build Random Forests learners for binary classification. After feature selection, we applied the Naïve Bayes classifier to the training data sets with the selected features, and then we used AUC to evaluate the performance of the classification. In total, 17 (ensembles) $\times$ 16 (data sets) $\times$ 10 (runs) $\times$ 5 (folds) = 13,600 models were built for the experiments

4.3 Results and Analysis

Experimental results are reported in Table 3. Each value presented in the table is the average over the ten runs of five-fold cross-validation outcomes. The best model for each data set is highlighted with **boldfaced** print. We also present the average performance (last row of the table) for each ensemble across all 16 data sets. The results demonstrate that the ensembles of two rankers, including EM-2IR (ensemble of IG and RF), EM-2RA (ensemble of RF and AUC-based feature selection) and EM-2RM (ensemble of RF and MI-based feature selection) outperformed the others for 12 out of 16 cases. The EM-2IR performed best on average performances across all 16 data sets. Due to space limitation, we didn't include the results for the four individual filters (IG, RF, AUC- and MI- based TBFS). The performances of individual filters are worse than the top performing ensemble techniques.

We also carried out a one-way Analysis of Variance (ANOVA) F test [2] on the AUC performance metric. The factor A represents the results from 17 different ensemble methods. In this ANOVA test, the results from all 16 data sets were taken into account together. A significance level of $\alpha = 5\%$ was used for all statistical tests. The ANOVA results are presented in Table 4. The test results indicate the classification performances of 17 different ensembles (Factor A) were significantly different from each other ($p = 0$).

Table 4. Analysis of Variance

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.20299	16	0.012687	5.8623	0
Error	5.8497	2703	0.002164		
Total	6.0527	2719			

We further conducted multiple pairwise comparison tests on Factor A. The multiple comparison test results are shown in Figure 1. The figure displays graphs with each group mean represented by a symbol ($\circ$) and the 95% confidence interval as a line around the symbol. Two means are significantly different if their intervals are disjoint, and are not significantly different if their intervals overlap. Matlab was used to perform the ANOVA and multiple comparisons presented in this work, and the assumptions for constructing ANOVA models were validated. Figure 1 shows that the two ensembles, EM-2IR and EM-2RM, performed best, followed by EM-2RA, EM-6O, EM-3IRM, EM-3IRA and EM-3RAM. EM-2IM, EM-2IA, EM-3IIAM, EM-2AM, and EM-11 performed significantly worse than the first two, EM-2IR and EM-2RM. Also there were no significant differences among EM-2IR, EM-2RM, EM-2RA, EM-6O, EM-3IRM, EM-3IRA and EM-3RAM. This indicates that ensembles of selected rankers (with very few rankers) are similar or better than ensembles of many or all rankers.

5 Conclusion

To alleviate the problems associated with a single feature selection method, ensemble techniques have been proposed. In this paper, we present a comprehensive study on the use of 17 different ensembles of rankers to select feature subsets. Our ensemble method combined a number of different rankings by averaging them and then selecting the highest ranked features.

In order to evaluate the effectiveness of each ensemble method, we applied ensemble feature selection techniques

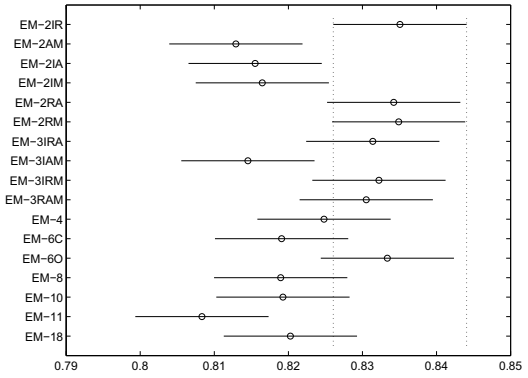

Figure 1. Multiple Comparison

to 16 software data sets. Classification models were built using the Naïve Bayes classifier with the selected features. These models are used to identify faulty software modules. We compared the 17 different ensembles of feature ranking techniques (built from 18 rankers) and conclude that no particular ensemble method dominates the others, but generally speaking, we can conclude that the ensembles of very few selected rankers are similar or better than ensembles of many or all rankers. The statistical significance of the results presented in this work is demonstrated through ANOVA models. Future work will involve conducting additional empirical studies with data from other software projects and application domains and experiments with other learners.

References

[1] W. Altidor, T. Khoshgoftaar, and J. Van Hulse. Classification with feature selection from noisy data: A comparative study of filters. *Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Technical Report*, May 2010.

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

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

[5] K. Gao, T. M. Khoshgoftaar, and H. Wang. An empirical investigation of filter attribute selection techniques for software quality classification. In *Proceedings of the 10th IEEE International Conference on Information Reuse and Integration*, pages 272–277, Las Vegas, Nevada, August 10-12 2009.

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

[7] 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.

[8] Y. Jiang, J. Lin, B. Cukic, and T. Menzies. Variance analysis in software fault prediction models. *Software Reliability Engineering, International Symposium on*, 0:99–108, 2009.

[9] T. M. Khoshgoftaar, M. Golawala, and J. Van 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.

[10] 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.

[11] A. G. Koru, D. Zhang, K. E. Emam, and H. Liu. An investigation into the functional form of the size-defect relationship for software modules. *IEEE Trans. Software Eng.*, 35(2):293–304, 2009.

[12] T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1):2–13, 2007.

[13] J. O. S. Olsson and D. W. Oard. Combining feature selectors for text classification. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 798–799, New York, NY, USA, 2006.

[14] R. L. Plackett. Karl pearson and the chi-squared test. *International Statistical Review*, 51(1):5972, 1983.

[15] 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.

[16] H. Wang, T. M. Khoshgoftaar, and K. Gao. Ensemble feature selection technique for software quality classification. In *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering*, pages 215–220, Redwood City, CA, USA, July 1-3 2010.

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

[18] C.-H. Yang, C.-C. Huang, K.-C. Wu, and H.-Y. Chang. A novel ga-taguchi-based feature selection method. In *IDEAL '08: Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, pages 112–119, Berlin, Heidelberg, 2008.

[19] T. Zimmermann, R. Premraj, and A. Zeller. Predicting defects for eclipse. In *ICSEW '07: Proceedings of the 29th International Conference on Software Engineering Workshops*, page 76, Washington, DC, USA, 2007. IEEE Computer Society.