The Effectiveness of the Fused Weighted Filter Feature Selection Method to Improve Software Fault Prediction

Fatemeh Alighardashi, Mohammad Ali Zare Chahooki
Engineering Faculty, Computer Engineering Department, Yazd University
Yazd, Iran
F.Alighardashi@stu.yazd.ac.ir, Chahooki@yazd.ac.ir

***Abstract*—improving the software product quality before releasing by periodic tests is one of the most expensive activities in software projects. Due to limited resources to modules test in software projects, it is important to identify fault-prone modules and use the test sources for fault prediction in these modules. Software fault predictors based on machine learning algorithms, are effective tools for identifying fault-prone modules. Extensive studies are being done in this field to find the connection between features of software modules, and their fault-prone. Some of features in predictive algorithms are ineffective and reduce the accuracy of prediction process. So, feature selection methods to increase performance of prediction models in fault-prone modules are widely used. In this study, we proposed a feature selection method for effective selection of features, by using combination of filter feature selection methods. In the proposed filter method, the combination of several filter feature selection methods presented as fused weighed filter method. Then, the proposed method caused convergence rate of feature selection as well as the accuracy improvement. The obtained results on NASA and PROMISE with ten datasets, indicates the effectiveness of proposed method in improvement of accuracy and convergence of software fault prediction.**

Keywords—Software fault prediction; Feature selection; Filter method; Machine learning

I. INTRODUCTION

Improving software product quality before releasing by periodic tests is one of the most expensive activities in software projects. Before system testing, identifying faults-prone components, improves effectiveness of efforts in software testing. The most faults in any software system are caused by a few of its components. The ‘‘80:20’’ rule in this area indicates that 20% of software modules may cause errors, cost and rework in the remaining 80% of them. So, it will be wasted little time in the whole production process by trying to eliminate error and improve the quality of a small part of software [1]. Due to limited resources in test activities of software projects, it is important to identify fault-prone modules and use the test resources for fault prediction in them. Software fault predictors based on machine learning algorithms, are effective tools for identifying faults-prone modules [2].

Binary classifier is the main model which classifies the software modules to two-class fault-prone and non-fault-prone ones. Ranking models, sort the modules based on their fault-prone probability. Therefore, the process of reliability improvement in software products can be done more effectively with an optimized allocation of the project resources to predict fault-prone modules.

So far, numerous machine learning methods for software fault prediction has been used [3, 4]. These methods include: Bayesian Network classifier [5, 6, 7], Discriminant Analysis [7], Logistic Regression [8, 9], C4.5 [7, 8], CART [7, 8], Neural Networks [7, 8, 10] and Support Vector Machine (SVM) [11, 12, 13].

Feature selection is one of the approaches to improve the accuracy and speed of machine learning algorithms [14]. In recent years, the feature selection is used in many applications of software engineering. Using of these methods has been growing in the field of software quality prediction, and especially in the software fault prediction [15, 16, 17, 18,19 20].

Research results of feature reduction have indicated that a sub-set of features can improve the accuracy of the machine learning algorithms. Combination of basic feature selection methods is the approach that recently has been suggested in this area.

In [21], the effect of the combination of feature selection methods and ensemble learning about the effectiveness of these methods is evaluated for predicting the fault in the software modules. The idea of the ensemble learning method has been proposed in this paper and for assessment of the impact of the feature selection methods on performance of the ensemble learning method also have been used from three methods such as greedy forward selection (GFS), Pearson’s correlation method and Fisher’s criterion (F-score).

In [22] represent a combination of the meta-heuristic optimization techniques (the genetic algorithms (GA) and the

particle swarm optimization (PSO)) and the Bagging methods for improving the accuracy of software fault prediction. The meta-heuristic optimization techniques are wrapper methods that these methods are used for feature selection, and the Bagging methods for tackling of the class unbalance problem are used. Using of the meta-heuristic optimization techniques are employed for increasing the ability of finding the highest quality solutions in the reasonable time frame. The purpose of this study of the above, selecting methods for feature selection is improvement of the accuracy. In other words, the speed factor is not the main purpose of this paper, so the filter methods have not been used in the feature selection process. The proposed idea of most studies on software fault prediction is limited to solving the problem of class unbalance or selecting the features individually. But this study combines two problems with each other and represents as a comprehensive approach.

In [23], the framework for classification of software modules which is named FECAR is presented. In this framework, from features clustering and ranking them for feature selection is used. Firstly the proposed method divides the basic features into k clusters based on correlation criterion, and then the most relevant features of each cluster are selected based on their relevancy. In this study the Symmetric Uncertainty method is used as the correlation measure. Also, three filter feature selection methods such as Information Gain (IG), Chi-Square (CS) and ReliefF (RF) have been used as a relevant measure.

In this paper, a combination of some of the filters features selection approaches are used to improve the accuracy and the convergence speed of the basic feature selection methods. The proposed method is a weighted combination of basic filter feature selection methods. Results show the effectiveness of the proposed method in improvement of accuracy and speed of software fault prediction.

The rest of the article is sectioned as follows: In Section 2, first, the feature selection methods are described. In Section 3, the proposed feature selection method is described. Dataset and evaluation criteria in this study are described in Section 4. Experimental implementation results of proposed method and comparison with other researches are given in Section 5. Finally, conclusions and future researches are described in Section 6.

II. Feature selection Methods

Feature selection, is one of the techniques that has been raised widely in machine learning. This method is very important in many applications, such as classification and regression. In the feature selection, we want to find the subset of features with the minimum possible size that is suitable for the learning process.

Feature selection methods are trying to find the best subset from a set with N features and 2N candidate of subsets. In all these methods, the subset is selected as the solution that could be optimized the value of evaluation function. Although each of the methods is trying to choose the best features, but the extent of the given possible solutions for finding the optimal solution is difficult and costly for medium and large values of N.

Feature selection is one of the approaches to improve the accuracy and speed of machine learning algorithms [24]. In recent years, many of the studies in feature selection on software fault prediction have been done.

A feature selection algorithm can be a combination of search techniques that provide a new subset of features with an evaluation criterion which score distinct subsets. The easiest method to test any proper subset of the features finds a subset that decreases the error rate [14].

In this paper, filter feature selection methods have been used to present proposed method. Filter feature selection methods have been used as a candidate criterion for scoring to a subset of features. These methods choose a subset of features without using any machine learning algorithm. In other words, in these methods, independent of machine learning algorithms, subsets of features have been selected by other concepts and competencies of them are evaluated. In these methods, according to the candidate criteria, each feature is scored, and based on scores, features are sorted as ascending, descending or random. In this paper, five filter methods such as Fisher Score, Gini Index, Kruskal Wallis, Minimum-Redundancy-Maximum-Relevance (mRmR) and Ttest from [25] have been used to present proposed combination filter method. These methods are briefly introduced In Table 1 and also the variables of these methods are briefly introduced In Table 2.

TABLE 1: FIVE FILTER METHODS HAVE BEEN USED TO PRESENT PROPOSED METHOD

Filter methods	Equations
Fisher Score	$FisherScore(f_i)=\frac{\sum_{j=1}^{c} n_j(\mu_{i,j}-\mu_i)^2}{\sum_{j=1}^{c} n_j\sigma_{i,j}^2}$
Gini Index	$GiniIndex(f)=1-\sum_{i=1}^{c}[p(i\mid f)]^2$
Kruskal Wallis	$K=\frac{12}{N(N+1)}\sum_{i=1}^{g} n_i(\bar{r}_i-\frac{N+1}{2})^2=\frac{12}{N(N+1)}\sum_{i=1}^{g} n_i(\bar{r}_i)^2-3(N+1)$
mRmR	Minimize Redundancy: $minW_I, W_I=\frac{1}{\lvert S\rvert^2}\sum_{i,j\in S} I(i,j)$ Maximize Relevance: $maxV_I, V_I=\frac{1}{\lvert S\rvert}\sum_{i\in S} I(h,i)$ $I(S_m;h)=\int\int p(S_m,h)\log\frac{p(S_m,h)}{p(S_m)p(h)}dS_m dh$
Ttest (t-score)	$R_t=\frac{\mu_i-\mu_j}{\sqrt{\frac{\sigma_i^2}{n_i}+\frac{\sigma_j^2}{n_j}}}$

TABLE 2: THE VARIABLES OF FILTER METHODS HAVE BEEN USED TO PRESENT PROPOSED COMBINATION FILTER METHOD

The variables	Description	The variables	Description
f	the feature	ni	number of observations in group 'i'
μ_i	the mean of the feature fi	rij	rank of observation 'j' in the group 'i'
nj	the number of samples in the jth class	$\overline{r}_i$	$\overline{r}_i = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i}$
$\mu_{i,j}$	the mean of fi on class j	r	the average rank of all the observations
$\sigma_{i,j}$	the variance of fi on class j	S	the set of features
c	the number of classes	I(i, j)	mutual information between features i and j
N	the total number of observations across all groups	h	target classes

III. PROPOSED METHOD

In the proposed method, feature selection is done by our fused weighted filter (WF) method. In different filter methods, according to the candidate criteria considered by them, scoring of features can be different. The score of each feature is actually featured weighs that it determines the position of them in the ranked list of features arranged by any filter method. In some filter methods, if the weight assigned to the features is more, it is diagnosed more relevant feature and is ranked higher in the list. In other filter methods higher position in the list of its features is to feature with less weight. Also, some filter methods are based on complex relationships for sorting a list of features based on their weight. Our proposed idea for fusing of filter methods is according to the following steps:

1) Features according to the priority of each feature in the list that is sorted by any of the filter methods, are re-weighted (higher priority in the list = more weight). The weights are then normalized.

2) Weight is calculated for each filter method based on their success in effectively prioritize features by evaluation score from training and validation samples. At this stage, we prioritize training database features by using of any of the filter methods and evaluate the accuracy of filter methods by the validate section. Then arrange methods based on their success rate and the method is the least accurate weight 0.1 is assigned. The other methods based on the difference in its accuracy over the previous methods, weighs "the weight difference divided by 10 and added to the previous method of weight" assigned. For example, the method that its accuracy is 5% more than the method that the weight is 0.1, weight 0.6 is assigned.

3) The weight of each feature multiplied by the weight of each filter method (new weighting to each of the features). Then the sum of the weights of features in each filter method is calculated and divided by the total number of filter methods. Therefore, if a feature is shown by x and W show also feature original weight. As well as the number of filter methods have M and $fm_j.W$ represents the weight of the j-th filter method. The final weight by the equation (1) is calculated.

$$x.NewW = \frac{\sum_{j=1}^{M} fm_j.W \times x.W}{M} \quad (1)$$

IV. EMPIRICAL LAYOUT

In this section, first we introduce the datasets that are discussed in this assessment. Then the criteria that reflect the accuracy of predicting fault in software modules are introduced. The criteria used in this study are selected so the results of this research are comparable with other studies in this area.

In evaluating the effectiveness of the proposed method for predicting the fault in software modules to set of datasets (ten datasets) consist of NASA and PROMISE are used. The datasets are freely available in the PROMISE Repository. In all of these datasets, the latest feature is the class label of each sample. In this feature, the value 0 represents module is not fault-prone and other value indicating the fault-prone. Table 3 indicates the characteristics of each dataset.

TABLE 3: CHARACTERISTICS OF PROMISE AND NASA DATASETS

Dataset	Version	Language	# of features	# of modules	% of defective modules
Ant	1.7	Java	20	745	22.3
Camel	1.6	Java	20	965	19.5
Jedit	4.3	Java	20	494	2.24
Lucene	2.4	Java	20	340	59.7
Tomcat	6.0	Java	20	858	9
CM1	-	C	21	498	9.83
MC2	-	C++	39	161	32.30
MW1	-	C	37	403	7.69
PC1	-	C	21	1109	6.94
PC2	-	C	37	778	2.05

The features of each dataset include different levels of granularity. NASA and PROMISE dataset, respectively, are in module and class level. All PROMISE samples include 20 features that are briefly described in Table 4. NASA samples also include a different number of features which are shown in Table 5.

TABLE 4: LIST OF FEATURES IN PROMISE DATASETS

Feature	Name	Description
x1	WMC	Weighted Methods per Class
x2	DIT	Depth of Inheritance Tree
x3	NOC	Number Of Children
x4	CBO	Coupling Between Object classes
x5	RFC	Response For a Class
x6	LCOM	Lack of Cohesion in Methods
x7	CA	Afferent Couplings

x8	CE	Efferent Couplings
x9	NPM	Number of public Methods
x10	LCOM3	Normalized version of LCOM
x11	LOC	Line Of Code
x12	DAM	Data Access Metric
x13	MOA	Measure Of Aggregation
x14	MFA	Measure of Functional Abstraction

x15	CAM	Cohesion Among Methods
x16	IC	Inheritance Coupling
x17	CBM	Coupling Between Methods
x18	AMC	Average Method Complexity
x19	MAX_CC	Maximum values of methods in the same class
x20	AVG_CC	Mean values of methods in the same class
-	Bug	Non-buggy or buggy

TABLE 5: LIST OF FEATURES IN NASA DATASETS

Feature	Description	CM1	MC2	MW1	PC1	PC2
x_1	Line count of code	×	×	×	×	×
x_2	McCable's cyclomatic complexity	×	×	×	×	×
x_3	McCable's essential complexity	×	×	×	×	×
x_4	McCable's design complexity	×	×	×	×	×
x_5	Total number of operators	×	×	×	×	×
x_6	Total number of operands	×	×	×	×	×
x_7	Number of unique operators	×	×	×	×	×
x_8	Number of unique operands	×	×	×	×	×
x_9	Number of unique operators and operands	×	×	×	×	×
x_{10}	Halstead's volume	×	×	×	×	×
x_{11}	Halstead's difficult	×	×	×	×	×
x_{12}	Halstead's length	×	×	×	×	×
x_{13}	Halstead's content	×	×	×	×	×
x_{14}	Halstead's effort	×	×	×	×	×
x_{15}	Halstead's error estimate	×	×	×	×	×
x_{16}	Halstead's programing time	×	×	×	×	×
x_{17}	Number of blank lines	×	×	×	×	×
x_{18}	Number of comment-only lines	×	×	×	×	×
x_{19}	Number of code-only lines	×			×	
x_{20}	Number of lines with both code and comments	×	×	×	×	×
x_{21}	Number of branches	×	×	×	×	×
x_{22}	Number of condition		×	×		×
x_{23}	Call pairs		×	×		×
x_{24}	Cyclomatic density		×	×		×
x_{25}	Number of decision		×	×		×
x_{26}	Decision density		×	×		×
x_{27}	Design density		×	×		×
x_{28}	Number of edge		×	×		×
x_{29}	Essential density		×	×		×
x_{30}	LOC executable		×	×		×
x_{31}	Number of parameter		×	×		×
x_{32}	Global data complexity		×			
x_{33}	Global data density		×			
x_{34}	Halstead's level		×	×		×
x_{35}	Maintenance severity		×	×		×
x_{36}	Number of modified condition		×	×		×
x_{37}	Number of multiple condition		×	×		×
x_{38}	Number of node		×	×		×
x_{39}	Normalized cyclomatic complexity		×	×		×
x_{40}	Percent comments		×	×		×
Number of features		21	39	37	21	37

One of the effective factors in comparison of the model with the other models is evaluated with the appropriate measures. Software fault prediction datasets are unbalance, the best evaluation criteria for assessment of the accuracy of the classifier for this type of datasets are AUC [26] and f-measure [11]. AUC criterion is given in Equation 2

$$AUC = \frac{1 + TP_r - FP_r}{2} \quad (2)$$

In this criterion, TP_r and FP_r are calculated as follows:

$$TP_r = \frac{TP}{TP + FN}$$

$$FP_r = \frac{FP}{FP + TN}$$

F-measure criterion is given in Equation 3.

$$F - measure = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (3)$$

In Equations 2 and 3, four criteria TP, FP, TN and FN are used. TP , is the number of the faulty modules, which is predicted correctly as fault-prone; FP , is the number of nonfaulty modules, that incorrectly have been predicted as fault-prone; TN , is the number of nonfaulty modules, which have predicted correctly as the nonfault-prone, and the FN also has a number of faulty modules, which incorrectly have been predicted as nonfault-prone.

V. Results and Discussion

In this section, first the results of the implementation of the proposed method on software fault prediction datasets will be represented. Then the results of this paper and the other recent studies in this area have been compared.

The datasets are used in this paper, first were divided into training data and test data (90 percent for training and 10 percent for the test). For selecting the optimal parameters of the proposed method 10-fold cross-validation method is used. Also for classifying the software modules into two classes fault-prone and none fault-prone, Smooth Twin Parametric-Margin SVM (STPMSVM) [27] method is employed.

Filter methods only return an ordered list of the most relevant features as output. So validating is required for selecting the top k feature of the list for all of the filter methods. So by using of training and validation data, the optimal value of k is calculated for each of the filter methods. Due to the optimal k, percentage of features that is chosen by each of the filter methods is shown in Table 6 and Table 7.

According to Table 6, the percentage of selected features of each datasets by Our proposed WF method is about 53% (by AUC criterion). The results represent that this method in comparison to five other filter methods (except of Fisher Score method) requires to the average of few features.

Table 6: Percentage of selected features by AUC criterion

Datasets	WF method	Filter Methods				
		Fisher Score	Gini Index	Kruskal Wallis	mRmR	Ttest
Ant	90	90	95	95	90	90
Camel	65	80	85	55	80	100
Jedit	10	65	75	15	65	100
Lucene	75	15	15	80	10	30
Tomcat	10	15	15	70	15	75
CM1	38.09	14.28	33.33	23.8	33.33	90.47
MC2	56.41	48.71	53.84	71.79	66.66	79.48
MW1	81.08	75.67	81.08	94.59	89.18	72.97
PC1	23.80	19.04	23.80	90.47	95.23	71.42
PC2	81.08	59.45	54.05	75.67	89.18	56.75
Average	**53.05**	**48.22**	53.11	67.13	63.36	76.61

Table 7: Percentage of selected features by F-measure criterion

Datasets	WF method	Filter Methods				
		Fisher Score	Gini Index	Kruskal Wallis	mRmR	Ttest
Ant	90	90	95	95	90	90
Camel	65	85	80	50	85	100
Jedit	10	65	75	15	65	100
Lucene	45	25	25	10	20	45
Tomcat	60	70	70	95	80	75
CM1	80.95	14.28	47.61	23.8	33.33	90.47
MC2	56.41	48.71	53.84	71.79	66.66	79.48
MW1	5.40	75.67	81.08	94.59	89.18	5.40
PC1	76.19	76.19	76.19	57.14	80.95	85.71
PC2	81.08	59.45	54.05	75.67	89.18	56.75
Average	**57.00**	60.93	65.78	58.80	69.93	72.78

The implementation results of five filter feature selection methods and the proposed fused weighted filter method on software fault prediction datasets is shown in table 8 and table 9. In last two rows of these tables, the average and standard deviation of each of the filter methods on all datasets are given. This comparison shows that the fused weighted filter method in the different datasets has better results rather than the other filter methods. The implementation results of the proposed method on software fault prediction datasets compare with using of all features are shown in table 10. The proposed method results graph is shown in Fig. 1.

Table 8: The result of proposed WF method and other Filter methods on ten datasets by AUC criteria

Datasets	WF method	Filter Methods				
		Fisher Score	Gini Index	Kruskal Wallis	mRmR	Ttest
Ant	0.88	0.88	0.88	0.88	0.88	0.88
Camel	0.70	0.69	0.64	0.60	0.76	0.66
Jedit	0.87	0.84	0.85	0.96	0.83	0.83
Lucene	0.66	0.66	0.57	0.66	0.45	0.51
Tomcat	0.85	0.76	0.76	0.83	0.88	0.82
CM1	0.80	0.87	0.84	0.83	0.90	0.87
MC2	0.83	0.78	0.83	0.74	0.80	0.83
MW1	0.91	0.89	0.91	0.89	0.91	0.89
PC1	0.87	0.77	0.80	0.82	0.89	0.76
PC2	0.97	0.98	0.96	0.97	0.97	0.92
Average	**0.83**	0.81	0.80	0.81	0.82	0.79
standard deviation	**0.09**	0.10	0.12	0.12	0.15	0.13

TABLE 9: THE RESULT OF PROPOSED WF METHOD AND OTHER FILTER METHODS ON TEN DATASETS BY F-MEASURE CRITERIA

Datasets	WF method	Filter Methods				
		Fisher Score	Gini Index	Kruskal Wallis	mRmR	Ttest
Ant	0.76	0.76	0.76	0.76	0.76	0.76
Camel	0.46	0.45	0.43	0.38	0.51	0.42
Jedit	0.14	0.12	0.13	0.33	0.11	0.11
Lucene	0.75	0.73	0.73	0.63	0.63	0.75
Tomcat	0.55	0.55	0.52	0.46	0.55	0.55
CM1	0.42	0.42	0.31	0.35	0.47	0.42
MC2	0.77	0.71	0.77	0.67	0.83	0.77
MW1	0.67	0.33	0.33	0.33	0.33	0.67
PC1	0.47	0.47	0.57	0.32	0.47	0.57
PC2	0.33	0.40	0.25	0.33	0.33	0.14
Average	**0.53**	0.49	0.48	0.45	0.49	0.51
standard deviation	**0.20**	0.20	0.23	**0.17**	0.21	0.24

TABLE 10: THE RESULT OF PROPOSED WF METHOD ON TEN DATASETS

Datasets	AUC		F-measure	
	All features	WF method	All features	WF method
Ant	0.82	**0.88**	0.70	**0.76**
Camel	0.66	**0.70**	0.42	**0.46**
Jedit	0.83	**0.87**	0.11	**0.14**
Lucene	0.61	**0.66**	0.60	**0.75**
Tomcat	0.80	**0.85**	0.52	**0.55**
CM1	0.64	**0.80**	0.22	**0.42**
MC2	0.81	**0.83**	0.70	**0.77**
MW1	0.89	**0.91**	0.46	**0.67**
PC1	0.81	**0.87**	**0.54**	0.47
PC2	0.91	**0.97**	0.33	**0.33**
Average	0.78	**0.83**	0.46	**0.53**

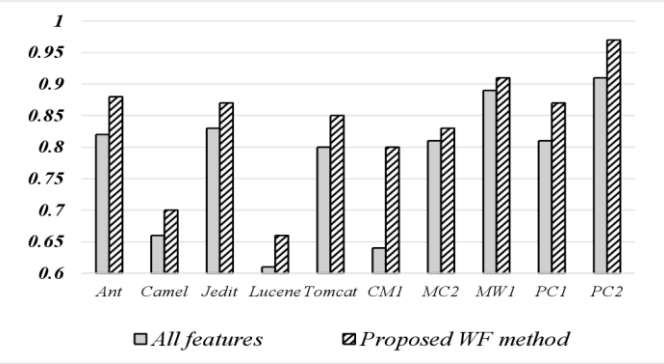

FIG 1: AUC CURVE DIAGRAM OF THE PROPOSED METHOD

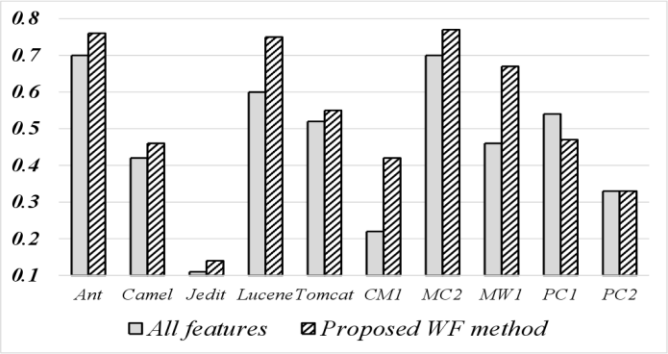

FIG 2: F-MEASURE CURVE DIAGRAM OF THE PROPOSED METHOD

Implementation of the proposed WF feature selection method on each dataset, causing a set of final features. According to experiment results, features x5 and x11 in the NASA dataset and features x1, x2, x6, x9, x18, x31 and x36 in the PROMISE dataset are the most effective features to improve the prediction accuracy of faults in software modules.

The results of this paper for PROMISE datasets in Table 11 and for NASA datasets in Table 12 has been compared with the results of other papers by AUC criteria. Also the results of this paper for NASA datasets in Table 13 have been compared with the results of other papers by F-measure criteria. In each paper, by several methods, better accuracy is reported. Finally, for each of dataset, the best accuracies are highlighted in the table 11, table 12 and table 13.

TABLE 11: COMPARE THE RESULT OF THIS PAPER WITH THE RESULT OF OTHER PAPERS ON PROMISE DATASETS BY AUC CRITERIA

Paper	Method	Ant	Camel	Jedit	Lucene	Tomcat
This paper	WF method	**0.88**	0.70	**0.87**	**0.66**	**0.85**
[6]	BNET	0.82	-	0.66	0.63	0.77
[21]	Enhanced APE	0.86	**0.80**	-	-	-
	Enhanced W-SVM	0.80	0.69	-	-	-
	Enhanced RF	0.82	0.72	-	-	-

TABLE 12: COMPARE THE RESULT OF THIS PAPER WITH THE RESULT OF OTHER PAPERS ON NASA DATASETS BY AUC CRITERIA

Paper	Method	CM1	MC2	MW1	PC1	PC2
This paper	WF method	**0.80**	**0.83**	**0.91**	**0.87**	**0.97**
[21]	Enhanced APE	-	-	-	-	0.95
	Enhanced W-SVM	-	-	-	-	0.94
	Enhanced RF	-	-	-	-	0.85
[22]	LR + PSO + Bag	0.74	0.78	0.75	0.85	0.83
	NB + PSO + Bag	0.76	0.73	0.75	0.79	0.82
	CART + PSO + Bag	0.61	0.68	0.68	0.83	0.79
	NN BP	0.71	0.71	0.62	0.78	0.92
[23]	NB + CIG	0.78	0.71	0.79	0.78	-
	NB + CRF	0.76	0.70	0.77	0.72	-
	NB + CFS	0.76	0.66	0.78	0.79	-
[28]	SVM	-	-	-	0.73	0.65
	CSSVM	-	-	-	0.81	0.75
	GA-CSSVM	-	-	-	0.83	0.80

TABLE 13: COMPARE THE RESULT OF THIS PAPER WITH THE RESULT OF OTHER PAPERS ON NASA DATASETS BY F-MEASURE CRITERIA

Paper	Method	CM1	MC2	MW1	PC1	PC2
This paper	Proposed WF method	**0.42**	**0.77**	**0.67**	**0.47**	**0.33**
[11]	DT	0.21	0.56	0.20	0.27	0.00
	KNN	0.13	0.57	0.23	0.43	0.00
	NB	0.24	0.50	0.00	0.33	0.05
	SVM	0.00	0.48	0.07	0.00	0.00
	R-SVM	0.35	0.57	0.46	0.39	0.01

The results shown that using the proposed WF method not only improves the speed of selecting features, But also has a significant impact on improving the accuracy of the fault prediction in the software modules.

VI. Conclusion and future work

In this paper due to the advantages and disadvantages of the filter feature selection methods, we use the combination of these methods to select the best features for predicting the fault of software modules. We achieved a suitable combined method for feature selection bdy the mixture of the feature selection methods. Then, this method is evaluated on the variety datasets of the software fault prediction. The results show the effectiveness of the used method. Therefore, our proposed WF method can find the best features with the highest speed for the improvement of the fault prediction accuracy. In this research, we use the five filter methods. In future we intend to work on the other combination of feature selection methods.

References

[1] G. Iker, "Applying machine learning to software fault-proneness prediction," J. Syst. Softw., vol. 81, no. 2, pp. 186–195, 2008.

[2] E. Rashid, S. Patnaik, and A. Usmani, "Machine Learning and Its Application in Software Fault Prediction with Similarity Measures," In Computational Vision and Robotics, pp. 37-45, Springer India, 2015.

[3] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Appl. Soft Comput., pp. 504–518, 2015.

[4] B. Ghotra, S. Mcintosh, and A. E. Hassan, "Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models," in Proc. of the 37th Int'l Conf. on Software Engineering (ICSE), 2015.

[5] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers," IEEE Trans. Softw. Eng., vol. 39, no. 2, pp. 237–257, 2013.

[6] A. Okutan, and O. Yıldız, "Software defect prediction using Bayesian networks," Empirical Software Engineering, vol. 19, no. 1, pp. 154-181, 2014.

[7] S. Lessmann, B. Baesens , C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: a proposed framework and novel findings," Softw. Eng. IEEE Trans., vol. 34, no. 4, pp. 485–496, 2008.

[8] R. Malhotra, and A. Jain, "Fault prediction using statistical and machine learning methods for improving software quality," JIPS, vol. 8, no. 2, pp. 241–262, 2012.

[9] A. Monden, T. Hayashi, S. Shinoda, K. Shirai, J. Yoshida, M. Barker, and K. Matsumoto, "Assessing the cost effectiveness of fault prediction in acceptance testing," Softw. Eng. IEEE Trans., vol. 39, no. 10, pp. 1345–1357, 2013.

[10] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," Expert Syst. Appl., vol. 37, no. 6, pp. 4537–4543, 2010.

[11] T. Choeikiwong and P. Vateekul, "Software Defect Prediction in Imbalanced Data Sets Using Unbiased Support Vector Machine," Inf. Sci. Appl. Springer Berlin Heidelb., vol. 339, pp. 923–931, 2015.

[12] A. H. Al-Jamimi, and L. Ghouti, "Efficient prediction of software fault proneness modules using support vector machines and probabilistic neural networks," Software Engineering (MySEC), 2011, 5th Malaysian Conference in. IEEE, 2011.

[13] H. Can, X. Jianchun, Z. Ruide, L. Juelong, Y. Qiliang, and X. Liqiang, "A new model for software defect prediction using particle swarm optimization and support vector machine," Control and Decision Conference (CCDC), 2013, 25th Chinese. IEEE, 2013.

[14] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Comput. Electr. Eng., vol. 40, no. 1, pp. 16–28, 2014.

[15] K. Muthukumaran, R. Akhila, and N. L. Murthy. "Impact of Feature Selection Techniques on Bug Prediction Models." Proceedings of the 8th India Software Engineering Conference. ACM, 2015.

[16] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim, "Reducing features to improve code change-based bug prediction". IEEE Transactions on Software Engineering, 39(4), 552-569, 2013.

[17] K. Gao, 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.

[18] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "Software measurement data reduction using ensemble techniques". Neurocomputing, 92, 124-132, 2012.

[19] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "A comparative study of ensemble feature selection techniques for software defect prediction". In Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on. IEEE, 135-140, 2010.

[20] A. Okutan, and O. T. Yıldız, "Software defect prediction using Bayesian networks." Empirical Software Engineering 19.1 (2014): 154-181.

[21] Laradji, I. H., Alshayeb, M., & Ghouti, L. "Software defect prediction using ensemble learning on selected features". Information and Software Technology, 58, 388-402, 2015.

[22] R. S. Wahono, N. Suryana, and S. Ahmad. "Metaheuristic Optimization based Feature Selection for Software Defect Prediction." Journal of Software 9.5 (2014): 1324-1333.

[23] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen, "Fecar: A feature selection framework for software defect prediction." Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual. IEEE, 2014.

[24] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Comput. Electr. Eng., vol. 40, no. 1, pp. 16–28, 2014.

[25] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, "Advancing feature selection research". ASU feature selection repository, 2010.

[26] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction". In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (p. 43). ACM, 2014.

[27] Z. Wang, Y. H. Shao, and T. R. Wu, "A GA-based model selection for smooth twin parametric-margin support vector machine". Pattern Recognition, 46(8), 2267-2277, 2013.

[28] B. Shuai, H. Li, M. Li, Q. Zhang, and C. Tang, "Software defect prediction using dynamic support vector machine," Comput. Intell. Secur. (CIS), 2013 9th Int. Conf., pp. 260-263, IEEE, 2013.