FSCR:A Feature Selection Method for Software Defect Prediction

Xiao Yu[1,2,3], Ziyi Ma[2,3], Chuanxiang Ma[2,3*] ,Yi Gu[2,3],Ruiqi Liu[4], Yan Zhang[2,3]
[1]State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, China
[2]School of Computer Science and Information Engineering, HuBei University, Wuhan, China
[3]Educational Informationalization Engineering Research Center of HuBei Province, Wuhan, China
[4] International School of Software, Wuhan University, Wuhan, China
*Corresponding author email: mxc838@hubu.edu.cn

***Abstract*—Prediction the number of faults in software modules can be more helpful instead of predicting the modules being faulty or non-faulty. Some regression models have been used for predicting the number of faults. However, the software defect data may involve irrelevant and redundant module features, which will degrade the performance of these regression models. To address such issue, this paper proposes a feature selection method based on Feature Spectral Clustering and feature Ranking (FSCR) for the number of software faults prediction. First, FSCR groups the original features with spectral clustering according to the correlation between every two features. Second, FSCR employs ReliefF algorithm to compute the relevance between each feature with respect to the number of faults and selects top *p* most relevant features from each resulted cluster. We evaluate our proposed method on 6 widely-studied project datasets with four performance metrics. Comparison with five existing feature selection methods demonstrates that FSCR is effective in selecting features for the number of faults prediction.**

Keywords—software fault prediction;regression model;feature selection; spectral clustering

I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. Based on the investigation of historical metrics, defect prediction aims to detect the defect proneness of new software modules. Therefore, defect prediction is often used to help to reasonably allocate limited development and maintenance resources [1]. So far, many efficient software defect prediction methods using statistical methods or machine learning techniques have been proposed [2-5], but they are usually confined to predicting a given software module being faulty or non-faulty by means of some binary classification techniques.

However, predicting the defect-prone of a given software module does not provide enough logistics to software testing in practice [6]. Some of the faulty software modules may have comparatively vast quantities of faults compared to other modules and hence require some additional maintenance resources to fix them. So, it may result in a waste of limited maintenance resources if simply predicting the defect-prone of a given software module and allocating the limited maintenance resources solely based on faulty and non-faulty information. If we are able to predict the accurate number of faults, software testers will pay particular attention to those software modules that have more number of faults, which makes testing processes more efficient in the case of limited development and maintenance resources. Thus, prediction the number of faults in software modules can be more helpful instead of predicting the modules being faulty or non-faulty [6].

A number of prior studies have investigated regression models on predicting the number of faults. Some researchers [7-12] have investigated genetic programming, decision tree regression, and multilayer perceptron in the context of the number of faults prediction and found that these models achieved good performance. Chen et al. [11] performed an empirical study on predicting the number of faults using six regression algorithms and found that the prediction model built with decision tree regression had the highest prediction accuracy in most cases. In another similar study, Rathore et al. [12] presented an experimental study to evaluate and compare the other six regression algorithms for the number of faults prediction. The results found that decision tree regression, multilayer perceptron, and linear regression achieved better performance in many cases.

However, the performance of these regression models is still vulnerable to irrelevant and redundant module features that may undermine the prediction effect. It is crucial to apply feature selection to the number of faults prediction since feature selection can filter out irrelevant and redundant features by evaluating the contributions of module features. The output of feature selection is a subset of the original feature set. This feature subset is more effective for the number of faults prediction.

In this paper, we propose a novel feature selection method, FSCR, to support feature selection for the number of faults prediction. FSCR is short for feature selection based on Feature Spectral Clustering and feature Ranking, which enhances feature selection for the number of software faults prediction via a two-stage approach. First, FSCR groups the original features with spectral clustering according to the correlation between every two features. Second, FSCR employs ReliefF algorithm to compute the relevance between the features and the number of faults and selects top *p* most relevant features from each resulted cluster.

We evaluate our proposed feature selection method, FSCR, by answering two research questions on performance. Experiments are conducted on 6 publicly available projects.

DOI reference number: 10.18293/SEKE2017-081

Experimental results show that FSCR can effectively select features to improve the performance of the models for the number of faults prediction.

II. Related Work

In this section, we first briefly review the existing defect prediction methods. Then, we review the existing feature selection methods.

A. Defect Prediction

Many researchers have proposed various models for predicting the module being faulty or non-faulty. Support vector machine [13-14], neural networks [15], decision trees [16] and Bayesian methods [17] paved the way for classification-based methods in the flied of defect prediction. These methods used software metrics to properly predict whether a module is defect-prone or not.

A number of prior studies have investigated regression models on predicting the number of software faults. Graves et al. [18] presented a generalized linear regression based method for the number of faults prediction using various change metrics datasets collected from a large telecommunication system and found that modules age, changes made to module and the age of the changes were significantly correlated with the defect-prone. Chen et al. [11] performed an empirical study on predicting the number of faults using six regression algorithms and found that the prediction model built with decision tree regression had the highest prediction accuracy in most cases. In another similar study, Rathore et al. [9] presented an experimental study to evaluate and compare the other six regression algorithms for the number of faults prediction. The results found that decision tree regression, genetic programming, multilayer perceptron, and linear regression achieved better performance in many cases. However, the prediction performance of these models gets worse when the defect datasets contain irrelevant and redundant features.

B. Feature Selection in Defect Prediction

A number of prior studies have investigated feature selection methods on predicting the module being faulty or non-faulty. Gao et al. [19] studied four different filter-based feature selection methods with five different classifiers on a large telecommunication system and found that the Kolmogorov-Smirnov method performed the best. Gao et al. [20] presented a comparative investigation to evaluate their proposed hybrid feature selection method, which first uses feature ranking to reduce the search space and then applies feature subset selection. In order to investigate different feature selection methods to classification-based bug prediction, Shivaji et al. [21] utilized six feature selection methods to iteratively remove irrelevant features until achieving the best performance of F-measure. Chen et al. [22] proposed a two-stage data preprocessing framework, TC, which combines feature selection and instance reduction. Liu et al. [23] proposed a new feature selection framework, FECAR, to conduct feature clustering and feature ranking.

III. Methodology

In this section, we present our FSCR method for the number of faults prediction. We first introduce the framework of our proposed method; then we present the detailed steps in the stage of feature spectral clustering and feature ranking.

A. The framework of our method

The method consists of two major stages: feature spectral clustering and feature ranking. Fig.1 illustrates the process of FSCR using a simple example.

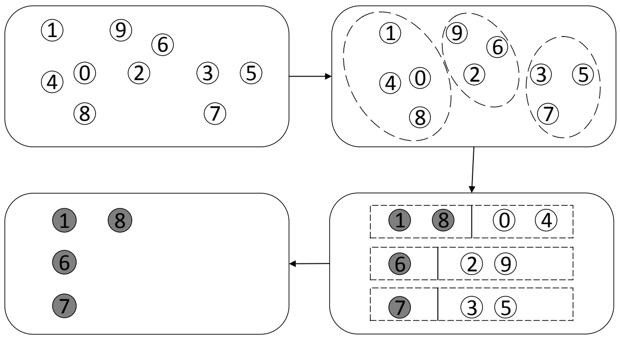

Figure 1. The process of FSCR

Assumes that the dataset has ten original features, represented by hollow circle in Fig.1. In the first stage, these features are partitioned into three clusters by using the spectral clustering algorithm, namely, C_1={0,1,4,8}, C_2={2,6,9} and C_3={3,5,7}. In the second stage, we rank all features in every clusters based on the relevance between each feature with respect to the number of software faults, and select the top p features from each cluster. Therefore, the final feature subset contains 1, 6, 7 and 8.

Therefore, the input of the FSCR method is the original feature set $\{f_1,f_2,\ldots,f_n\}$, the correlation measure FA between every two features, the relevance measure FB between each feature and the number of software faults, the number of the clusters k and the number of selected features m. The output of the FSCR method is the final feature subset R. The details are shown in the Algorithm 1.

B. The first stage

The first stage partitions the original features into k clusters such that features in the same cluster are similar and features in different clusters are dissimilar to each other. The main goal of the stage of feature clustering is to eliminate redundant features that have similar effect with other features. Note that in contrast to traditional clustering, our goal is to group features rather than instances.

This stage first uses the Pearson correlation coefficient to calculate the pairwise correlation between every two features using the following formula:

$$r=\frac{\sum_i(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i(x_i-\bar{x})^2}\sqrt{\sum_i(y_i-\bar{y})^2}} \tag{1}$$

where values x_i and y_i denote the numeric values of the feature x and feature y in the i-th instance (i=1,2,…,n), $\bar{x}=\frac{1}{n}\sum_{k=1}^{n}x_i$ and $\bar{y}=\frac{1}{n}\sum_{k=1}^{n}y_i(i=1,2,\ldots,n)$.

Algorithm 1. FSCR method

```
Input:
    Original feature set {$f_1, f_2, \ldots, f_n$}
    Correlation measure FA between every two features
    Relevance measure FB between each feature and the number of software faults
    Number of the clusters $k$
    Number of selected features $m$
Output:
    Final feature subset $R$
/*The first stage: feature clustering*/
1: for $i$=1 to $n$ do
2:     for $j$=1 to $n$ do
3:         Compute the correlation between $f_i$ and $f_j$ using FA;
4:     end for
5: end for
6: Partition original $n$ features into $k$ clusters {$C_1, C_2, \ldots, C_k$}using spectral clustering algorithm;
/*The second stage: feature ranking*/
7: for $i$=1 to $n$ do
8:     Using the relevance measure FB to compute the relevance between $f_i$ and the number of the software faults;
9: end for
10: for $i$=1 to $n$ do
11:     Ranking the features in $C_i$ in descending order according to the relevance;
12: end for
13: for $i$=1 to $k$ do
14:     Adding top $\left\lceil \frac{|C_i| \times m}{n} \right\rceil$ features of $C_i$ into $R$;
15: end for
16: return R ;
```

Then, this stage uses spectral clustering to cluster the original feature set based on the correlation between every two features. Different from the other distance-based clustering algorithms, spectral clustering [24] makes use of the spectrum (eigenvalues) of the similarity matrix of the instances to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix can be defined as a symmetric matrix W, where W_{ij} represents a measure of the similarity between every two instances X_i and X_j.

C. *The second stage*

In this stage, we select top p relevant features from each resulted cluster to construct the final feature subsets. We first employ the ReliefF algorithm [25] to compute the relevance between each feature and the number of the software faults. ReliefF randomly selects an instance R_i, but then searches for k of its nearest neighbors from the same class, called nearest hits H_j, and also k nearest neighbors from each of the different classes, called nearest misses M_j(C). It updates the quality estimation for all features depending on their values for R_i, hits H_j and misses M_j(C). If instances R_i and H have different values of the attribute A then the attribute A separates two instances with the same class which is not desirable so we decrease the quality estimation W[A]. On the other hand if instances R_i and M have different values of the attribute A, then the attribute A separates two instances with different class values which is desirable so we increase the quality estimation W[A]. The whole process is repeated for q times, where q is a user-defined parameter. In this experiment, we use the default parameter specified by *sklearn* [26].

Then, we rank the features in C_i in descending order according to the relevance and select $\left\lceil \frac{|C_i| \times m}{n} \right\rceil$ features from each clusters, where $|C_i|$ is the number of the features in the cluster C_i, m is the size of the final feature subset and n represents the number of the original features. The selected features construct the final feature subset. According to literature [19], we select $\lceil \log_2 n \rceil$ features from the original features.

IV. Experiment Setup

A. *Data set*

In this experiment, we employ 6 available and commonly used software project datasets with their 22 releases which can be obtained from PROMISE [27]. The details about the datasets is shown in Table I, where *#Instance* represents the number of instances, *#Defects* represents the total number of faults in the release, *%Defect* represents the percentage of defect-prone instances, and *Max* is the maximum value of faults. There are the same 20 independent variables (the 20 feature metrics) and one dependent variable (the number of faults) in the six datasets. A comprehensive list of the metrics refers to literature [12].

TABLE I. Details of Experiment Dataset

Project	*Release*	*#Instance*	*#Defects*	*%Defects*	*Max*
Ant	Ant-1.3	125	33	16.0%	3
	Ant-1.4	178	47	22.5%	3
	Ant-1.5	293	35	10.9%	2
	Ant-1.6	351	184	26.2%	10
	Ant-1.7	745	338	22.3%	10
Camel	Camel-1.0	339	14	3.4%	2
	Camel-1.2	608	522	35.5%	28
	Camel-1.4	872	335	16.6%	17
	Camel-1.6	965	500	19.5%	28
Jedit	Jedit-3.2	272	382	33.1%	45
	Jedit-4.0	306	226	24.5%	23
	Jedit-4.1	312	217	25.3%	17
	Jedit-4.2	267	106	13.1%	10
	Jedit-4.3	492	12	2.2%	2

Project	*Release*	*#Instance*	*#Defects*	*%Defects*	*Max*
Synapse	Synapse-1.0	157	21	10.2%	4
	Synapse-1.1	222	99	27.0%	7
	Synapse-1.2	256	145	33.6%	9
Xalan	Xalan-2.4	724	111	15.3%	7
	Xalan-2.5	804	388	48.3%	9
	Xalan-2.6	886	412	46.5%	6
Xerces	Xerces-1.3	503	69	13.7%	30
	Xerces-1.4	589	438	74.4%	62

B. *Performance measures*

Since CERFS is a model to predict the number of faults, it should be evaluated using criteria for regression models. In the experiment, we employ root mean square error (RMSE) to measure the performance. In addition, considering the imbalanced characteristic of software defect datasets, we also employ three commonly used performance measures that evaluate classification models, including pd, pf and G-measure. These performance measures are defined in Table III and summarized as follows.

TABLE II. PERFORMANCE MEASURES

		Actual	
		yes	*no*
Predicted	*yes*	*TP*	*FP*
	no	*FN*	*TN*
pd	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
G-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		
RMSE	$\sqrt{\frac{\sum_{i=1}^{n}\lvert\bar{Y_i}-Y_i\rvert^2}{n}}$		

● Probability of detection or pd is the measure of defective modules that are correctly predicted within the defective class. The higher the pd, the fewer the false negative results.

● Probability of false alarm or pf is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike pd, the lower the pf value, the better the results.

● G-measure is a trade-off measure that balances the performance between pd and pf. A good prediction model should have high pd and low pf, and thus leading to a high G-measure.

● RMSE measures the deviation between the predicted value $\bar{Y}_i$ and the actual value Y_i. It is a good measure of accuracy to compare prediction errors of different regression models for a given variable, e.g., the number of faults.

C. *Research Questions*

Our evaluation answers two research questions.

RQ1. Does our proposed FSCR method perform better than state-of-the art feature selection methods in terms of predicting the modules being faulty or non-faulty?

This question validates the important criterion of defect prediction: the performance improvement in terms of pd, pf and G-measure (as defined in Section IV-B).

RQ2. Does our proposed FSCR approach perform better than state-of-the art feature selection methods in terms of the accuracy of predicting the number of the software faults?

This question validates the important criterion of the number of faults prediction: the performance improvement in terms of RMSE. (as defined in Section IV-B).

We compare our method with six classical feature selection methods in defect prediction:(1)Full, (2)Chi-Square [28], (3) Signal-to-Noise [29], (4)Information Gain [30], (5)Gain Ratio [31], and (6)FSCAR [23].

Full is the original feature subset. Compared to this method, we can study whether the FSCR can improve the performance of the number of faults prediction. Chi-Square (CS) and Signal-to-Noise (S2N) are statistic-based feature selection methods. Information Gain (IG) and Gain Ratio (GR) is the probability-based feature selection method. FECAR is a feature selection method combining feature ranking and feature clustering proposed by Liu et al [23]. FECAR first clusters features via k-medoids method and then select several representative features from each cluster.

D. *Experiment Procedure*

In experiments, we performed 10-fold cross validation when training classifiers on the selected features throughout this paper, to avoid any potential problem of overfitting particular training and test sets within a specific project. In 10-fold cross validation, a dataset is divided into 10 folds at random. Nine of the ten folds take turns to be used as the training set while the other fold is used as the test set. The training data are used to build a regression model; then the built model is evaluated on the test data. The above procedure is repeated 300 times (10 folds 30 independent runs) in total for each feature selection method to avoid sample bias. Then, the mean values of performance for all methods are calculated.

In order to compare the performance of feature selection methods, we employ three regression models in defect prediction, Bayesian Ridge Regression (BRR), Gradient Boosting Regression (GBR) and Linear Regression (LR). The reason we choose these regression models is that these models perform best in predicting the number of software faults [11-12].

V. EXPERIMENT RESULTS

In this section, we present the experiment results to answer our two research questions mentioned above.

A. *RQ1*

As mentioned in Sections IV-C, we compare our method FSCR with six feature selection methods. Table IV records the pd, pf and G-meausre of six datasets with six different feature selection methods on three regression models, BRR, GBR, LR. The column "Full" presents the training set without involving any feature selection method; W/D/L, short for Win/Draw/Loss, denotes the number of projects, on which FSCR performs better than, the same as, or worse than another method, in terms of G-measure.

As is shown in the Table III, FSCR performs better G-measure values than all the other methods. For BRR model, FSCR achieves the best average pd and G-measure value, but fails in the best pf value. For GBR model, FSCR can achieve the best pf and G-measure values. For LR model, FSCR achieves best values in terms of all the three measures. The Win/Draw/Loss values shows that, on three regression models, FSCR outperforms others on over half of projects in terms of all the three measures.

TABLE III. AVERAGE PERFORMANCE OF 6 PROJECTS WITH THREE REGRESSION MODEL ON PD, PF, AND G-MEASURE

Model	Metric	Full	FSCR	CS	GR	S2N	IG	FECAR
BRR	PD	0.512	**0.584**	0.514	0.521	0.556	0.548	0.579
	PF	0.236	0.169	0.182	0.171	**0.164**	0.165	0.170
	G	0.613	**0.668**	0.591	0.586	0.642	0.622	0.665
	W/D/L	4/0/2		5/0/1	4/0/2	5/0/1	4/0/2	4/0/2
GBR	PD	0.479	0.521	0.466	0.498	**0.535**	0.501	0.513
	PF	0.157	**0.121**	0.206	0.142	0.123	0.161	0.126
	G	0.610	**0.637**	0.585	0.613	0.633	0.609	0.631
	W/D/L	6/0/0		5/0/1	4/0/2	6/0/0	4/0/2	4/1/1
LR	PD	0.504	**0.591**	0.434	0.536	0.523	0.511	0.586
	PF	0.244	**0.152**	0.183	0.213	0.221	0.175	0.159
	G	0.604	**0.671**	0.565	0.625	0.609	0.649	0.668
	W/D/L	5/0/1		4/0/2	4/0/2	6/0/0	6/0/0	3/0/3

Fig. 2 shows the box-plots of G-measure values, with six methods for three regression models on 6 projects. For BRR model, the median value by FSCR is much higher than that by all the other methods. For GBR model, the median value by FSCR is higher than that by CS and S2N, while is similar with that by GR and IG, and is a little lower than that by FECAR. However, the maximum by FSCR is much higher than FECAR and all the other methods. For LR model, the median is similar with that by GR and FECAR, while is much higher than that by S2N and IG. In addition, the maximum by FSCR is much higher than that by all the other methods.

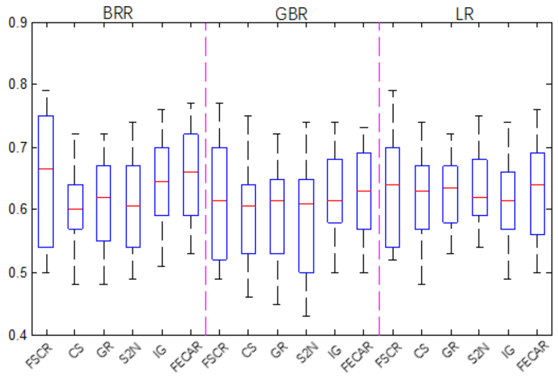

Figure 2. Box-plots for G-measure on 6 projects with three regression models.

RQ1 Summary. According to the experiment results in Table 4 and Figure 2, we conclude that FSCR can perform better than state-of-the-art feature selection methods in terms of predicting the modules being faulty or non-faulty.

B. RQ2

Tables IV, V and VI present the detailed RMSE values of each project on three regression models. From these tables, we can observe that FSCR performs better average RMSE value than all the other methods. The Win/Draw/Loss records also indicate that FSCR wins other methods on most projects on three regression models in term of RMSE measure. In addition, Hedges'g [32] is employed to demonstrate the effect size. The effect size of Hedges'g values are greater than 1.0 on most projects, which can be interpreted as a large improvement.

TABLE IV. RMSE VALUES ON 6 PROJECTS USING BAYESIAN RIDGE REGRESSION WITH THE HEDGES'G

Project	Full	FSCR	CS	GR	S2N	IG	FECAR
Ant	1.155	**0.829**	1.074	1.129	0.945	1.176	0.921
Camel	1.046	1.031	1.142	1.023	0.972	**0.824**	0.837
Jedit	1.426	0.986	0.965	0.965	1.028	0.975	**0.912**
Synapse	1.247	**0.892**	0.978	0.911	0.945	0.994	0.978
Xalan	1.010	**0.714**	1.123	1.101	0.897	1.109	0.925
Xerces	1.206	**0.821**	0.912	0.956	0.852	0.912	0.944
AVG	1.181	**0.878**	1.032	1.014	0.939	0.998	0.919
W/D/L	6/0/0		6/0/0	5/0/1	5/0/1	5/0/1	5/0/1
Hedges'g	**2.252**		**1.457**	**1.327**	0.657	0.980	0.462

TABLE V. RMSE VALUES ON 6 PROJECTS USING GRADIENT BOOSTING REGRESSION WITH THE HEDGES'G

Project	Full	FSCR	CS	GR	S2N	IG	FECAR
Ant	1.011	0.986	**0.894**	0.954	0.949	1.024	0.929
Camel	0.945	1.031	1.035	**0.927**	0.975	0.961	1.163
Jedit	1.295	**0.714**	0.917	1.082	1.021	0.913	1.075
Synapse	1.091	**0.821**	0.941	0.959	1.047	0.974	0.838
Xalan	0.906	**0.837**	1.109	1.056	1.145	1.127	0.914
Xerces	1.472	**0.892**	0.952	1.214	0.969	0.917	0.977
AVG	1.120	**0.880**	0.974	1.032	1.017	0.986	0.982
W/D/L	5/0/1		6/0/0	4/0/2	4/0/2	5/0/1	6/0/0
Hedges'g	**1.361**		0.939	**1.356**	**1.420**	**1.064**	0.873

TABLE VI. RMSE VALUES ON 6 PROJECTS USING LINEAR REGRESSION WITH THE HEDGES'G

Project	Full	FSCR	CS	GR	S2N	IG	FECAR
Ant	1.152	**0.957**	0.982	1.053	0.964	1.058	0.973
Camel	1.059	1.045	1.123	0.949	1.103	**0.934**	1.078
Jedit	0.914	**0.794**	0.994	1.027	1.025	0.853	0.935
Synapse	1.015	1.124	0.854	1.154	1.161	**0.927**	0.926
Xalan	1.205	**0.885**	0.942	1.048	1.054	1.185	0.910
Xerces	1.012	**0.921**	0.952	1.038	1.035	0.924	0.953
AVG	1.059	**0.954**	0.974	1.044	1.057	0.980	0.962
W/D/L	5/0/1		5/0/1	5/0/1	6/0/0	5/0/1	5/0/1
Hedges'g	0.944		0.193	**1.048**	**1.075**	0.219	0.086

RQ2 Summary. According to the experiment results in Tables 4-6, we conclude that FSCR can perform better than state-of-the-art feature selection methods in terms of the number of faults prediction.

VI. Conclusion and Future Work

In this paper, we propose a novel feature selection method for the number of faults prediction. The method involves the following two stages: in the first stage, we employ a feature spectral clustering method to cluster the original features; in the second stage, we select the highly relevant features from each cluster. Experiments on 6 project datasets indicate that the proposed method, FSCR, can perform competitive results for the number of faults prediction.

In the future, we will further investigate the impact of the parameters setting, such as the number of clusters and the number of features selected from each cluster. In addition, we would like to validate the generalization ability of our method on more datasets [33-34].

Acknowledgment

This work is partly supported by Educational Informationalization Engineering Research Center of HuBei Province.

References

[1] F. Rahman, D. Posnett, P. Devanbu , Recalling the imprecision of cross-project defect prediction, Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering,2012, 61.

[2] Q. Song, Z. Jia, M. Shepperd, et al, A general software defect proneness prediction framework, Software Engineering, IEEE Transactions on, 2011, 37(3): 356-370.

[3] X. Yang, K. Tang, X. Yao, A Learning-to-Rank Approach to Software Defect Prediction, IEEE Transactions on Reliability, 2015,64(1): 234-246.

[4] R. Malhotra, A systematic review of machine learning techniques for software fault prediction, Applied Soft Computing, 2015, 27: 504-518.

[5] M. Shepperd, D. Bowes, T. Hall, Researcher bias: The use of machine learning in software defect prediction, IEEE Transactions on Software Engineering, 2014, 40(6): 603-616.

[6] N. E. Fenton, M. Neil, A critique of software defect prediction models, IEEE Transactions on software engineering, 1999, 25(5): 675-689.

[7] Rathore S S, Kuamr S, Comparative analysis of neural network and genetic programming for number of software faults prediction, 2015 National Conference on Recent Advances in Electronics & Computer Engineering (RAECE), 2015: 328-332.

[8] W. Afzal, R. Torkar, R.Feldt, Prediction of fault count data using genetic programming, Multitopic Conference, 2008. INMIC 2008. IEEE International. IEEE, 2008: 349-356.

[9] S. S. Rathore, S. Kumar , Predicting number of faults in software system using genetic programming, Procedia Computer Science, 2015, 62: 303-311.

[10] S. S. Rathore, S.Kumar, A Decision Tree Regression based Approach for the Number of Software Faults Prediction,” ACM SIGSOFT Software Engineering Notes, 2016, 41(1): 1-6.

[11] M. Chen, Y. Ma, An empirical study on predicting defect numbers, 28th International Conference on Software Engineering and Knowledge Engineering, 2015: 397-402.

[12] S. S. Rathore, S. Kumar, An empirical study of some software fault prediction techniques for the number of faults prediction, Soft Computing, 2016: 1-18.

[13] D. Gray, D. Bowes, N. Davey, et al, Using the support vector machine as a classification method for software defect prediction with static code metrics, International Conference on Engineering Applications of Neural Networks. Springer Berlin Heidelberg, 2009: 223-234.

[14] Z. Yan, X. Chen, P. Guo, Software defect prediction using fuzzy support vector regression, International Symposium on Neural Networks. Springer Berlin Heidelberg, 2010: 17-24.

[15] M. M. T. Thwin, T. S.Quah, Application of neural networks for software quality prediction using object-oriented metrics, Journal of systems and software, 2005, 76(2): 147-156.

[16] J. Wang, B. Shen, Y.Chen, Compressed C4. 5 models for software defect prediction, 2012 12th International Conference on Quality Software. IEEE, 2012: 13-16.

[17] T. Wang, W. Li, Naive bayes software defect prediction model, Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on. IEEE, 2010: 1-4.

[18] T. L. Graves, A. F. Karr, J. S. Marron, et al, Predicting fault incidence using software change history, IEEE Transactions on software engineering, 2000, 26(7): 653-661.

[19] K. Gao, T. M. Khoshgoftaar, H. Wang. An empirical investigation of filter attribute selection techniques for software quality classification. Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on. 272-277. IEEE, 2009.

[20] K. Gao, T.M. Khoshgoftaar, H. Wang, et al. Choosing software metrics for defect prediction: an investigation on feature selection techniques . Software Practice & Experience, 41(5):579-606, 2011.

[21] S. Shivaji, J. E. J. Whitehead, R. Akella, et al. Reducing features to improve bug prediction. Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 600-604, 2009.

[22] J. Chen, S. Liu, W. Liu, et al. A Two-Stage Data Preprocessing Approach for Software Defect prediction. Software Security and Reliability (SERE), 2014 Eighth International Conference on. 20 - 29. IEEE, 2014.

[23] S. Liu, X. Chen, W. Liu, et al. FECAR: A Feature Selection Framework for Software Defect Prediction. 2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC). IEEE Computer Society, 426-435, 2014.

[24] Von Luxburg U. A tutorial on spectral clustering. Statistics and computing, 2007, 17(4): 395-416.

[25] Robnik-Šikonja M, Kononenko I. Theoretical and empirical analysis of ReliefF and RReliefF. Machine learning, 2003, 53(1-2): 23-69.

[26] http://scikit-learn.org/

[27] G. Boetticher, T. Menzies, T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007 <http://promisedata.org/repository>.

[28] Jin X, Xu A, Bie R, et al. Machine learning techniques and chi-square feature selection for cancer classification using SAGE gene expression profiles. International Workshop on Data Mining for Biomedical Applications. Springer Berlin Heidelberg, 2006: 106-115.

[29] Plapous C, Marro C, Scalart P. Reliable A posteriori Signal-to-Noise Ratio features selection. 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. IEEE, 2005: 66-69.

[30] Abadi D. Perbandingan Algoritme Feature selection information gain dan Symmetrical Uncertainty pada Data Ketahanan Pangan [J]. UT - Computer Science, 2013.

[31] Praveena P R, Valarmathi M L, Sivakumari S. Gain ratio patio based feature selection method for privacy preservation [J]. Ictact Journal on Soft Computing, 2011, 1(4).

[32] Kampenes, V. By, et al, A systematic review of effect size in software engineering experiments, Inform. Softw. Technol. 49.11 (2007) 1073-1086.

[33] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[34] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.