International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I4P216

Volume 70 Issue 4, 188-202, April 2022
© 2022 Seventh Sense Research Group®

Original Article

Random Permutation-based Hybrid Feature Selection for Software Bug Prediction using Bayesian Statistical Validation

Tamanna[1], Om Prakash Sangwan[2]

[1,2]*Department of Computer Science and Engineering, GJUST, Hisar, India*

[1]tamannasharma100@gmail.com

Received: 12 February 2022 Revised: 16 March 2022 Accepted: 18 March 2022 Published: 26 April 2022

Abstract - *Software Fault Prediction (SFP) is a key practice in developing quality software. To cater to rising human expectations, the software is getting complex and increasing source code size (adding new functionalities). A strategy like SFP can help detect faults beforehand and avoid software downtime. To reduce the cost of SFP, we propose a Permutation-based hybrid feature selection model (PFS). This model helps remove irrelevant and redundant features without compromising classifier performance. PFS has been compared with five different supervised feature selection methods – Chi-squared, Correlation, Sequential Forward Feature Selection, Sequential Backward Feature Selection, and Mutual Information. Random Forest (RF) classifier is employed, and experimental results (Accuracy, Precision, Recall, and AUC-ROC) were found on Twenty-four different datasets of three public software repositories. Bayesian statistical analysis of AUC-ROC results was carried out, and it was found that PFS was able to outperform other techniques by lower computational time and lower dimensions.*

Keywords - *Feature selection, Bayesian signed-rank test, ROC-AUC, Fault prediction.*

1. Introduction

Software quality plays a key role in software engineering. Quality has a direct relation with the frequency of bugs in software. The gap between desired and incorrect behaviour of the software is known as a bug. The increasing complexity of software needs a mechanism that talks about bug proneness of a module (class, method, etc.). This mechanism is termed software bug prediction. It has been an active field of software engineering for many years. Various soft computing techniques (machine learning, fuzzy logic, hybrid methods, etc.) are employed and explored for software bug prediction. Bug prediction model with the least complexity and better Accuracy can be the best choice for Software Practitioners [1]. The performance of all the computational intelligence (like machine learning etc.) techniques depend hugely on the choice of features employed in the training of models. Feature selection filters out the noisy, missing, and redundant features from the feature set [2]. In this study, we propose an SFP employing Permutation-based Hybrid Feature Selection (PFS), a combination of filter and wrapper methods. To validate its performance and efficacy, we compared its performance with both filter (Correlation, Chi-squared and Mutual Information) and wrapper methods (Sequential Forward Feature Selection and Sequential Backward Feature Selection) by using Random Forest Classifier [25], [26]. This comparison has been statistically validated with Bayesian Signed Rank Test [11]. In PFS, we first rank the available features with the help of permutation (random shuffling) and then filter out the features that contribute to the classification model's performance decrease. Twenty-four different datasets from AEEEM, NASA, and PROMISE software engineering repository are used for experimental study. The aim of this research article is to answer the following research questions:

- RQ 1. How have feature selection techniques been performed?
- RQ 2 What insights can be drawn from Bayesian Statistics?
- RQ 3 Best feature selection technique amongst the considering techniques?

The structure of this research article is: Section 2 surveys related work, and Section 3 explains the methodology. Section 4 discusses results, and the conclusion is presented in Section 5.

2. Literature Survey

This section explains previous studies in software bug prediction and usage of feature selection methods in software bug prediction. Bilal et al. [4] implemented various machine learning techniques-Multiple Layer Perceptron, Support

Vector Machine, Decision Tree, Radial Basis Function, Random Forest, Hidden Markov model, Credal Decision tree, k-nearest neighbour, average one dependency estimator, and Naïve Bayes on seven broadly used datasets. Random forest shows the best performance, followed by SVM in Accuracy.

Menzies et al. [5] did a comparative analysis with Naive Bayes and Rule Induction on ten different MDP NASA datasets. This study claimed that the machine learning algorithm selection is much more important than the attributes used for prediction. While another study by Song et al. [6] proposed a model (framework) for unbiased software bug prediction and challenges that results of Menzies [5] baseline approach is biased due to problem in attribute selection criteria. Different learning schemes are evaluated, and the best one is selected for bug prediction on NASA and Promise repository dataset. Statistical experiments show that no learning scheme is best for all the datasets and different combinations of attribute selectors and learning algorithms yield different performances. Agarwal and Tomar [7] say that "bug prediction is crucial and indirect task depends on software metrics". Linear Twin Support Vector Machine-based model was proposed in association with F-Score-based feature selection. Experimental results on four datasets of the Promise repository show that a reduced set of features could help increase the prediction capability of the learning model. Liu et al. [8] proposed a feature clustering, and feature ranking model which divides features into partitions based on Symmetric Uncertainty and then relevance set (based on Information gain, Chi-Square, and RelieF) of features are selected from each partition. Results of empirical studies in this model show that not only irrelevant but redundant features also increase the complexity (time consumption, cost, etc.) of learning algorithms.

Khoshgoftaar et al. [9] addressed twin issues of high dimensionality and data unbalancing in classification algorithms and employed data sampling with eighteen feature selection techniques. Three different real-world datasets (with varied data balance) are used for experimental studies, and results conclude iterative method performs better than non–iterative methods. Balogun et al. [10] address the relation of high dimensionality with the performance of the software bug prediction model. This study's motivation is that the heart of the feature selection method is the choice of search method. Therefore, different feature selection technique results in a different subset of features. Filter feature subset and filter feature ranking methods were employed in association with four learning algorithms on five different NASA datasets. Information gain among filter feature ranking and best first search among filter feature subset selection methods shows best results and concluded that no best universal feature selection method exists. Performance changes with different datasets and choice of software bug prediction algorithm. Catal et al. [11] implemented nine classifiers, including machine learning techniques in association with algorithms based on artificial immune systems (AIS). Also, they investigated the effect of metrics set, dataset type and size, etc. They used NASA datasets for experiments and concluded that Random forest outperforms larger datasets and Naïve Bayes for smaller datasets based on AUC-ROC. AIS-based algorithms work best when method-level metrics are employed. Jakhar et al. [12] also used NASA datasets (CM1, MC1, MC2, PC3, PC4) for finding the significant set of features by ranking based on the combined weight of seven feature selection techniques on eight classifiers (Naïve Bayes, J48, LibSVM, Multilayer Perceptron, K-star, and JRip) and shows improved results.

3. Methodology

This experimental study aims to find a definitive outperforming Feature Selection technique based on Bayesian Statistical Analysis. The flowchart of the experimental setup is depicted in Figure 1. The experiment has been carried out in Python language employing the sci-kit-learn library [31]. Bayesian analysis has been done using two packages – Autorank[32] and Baycomp[14].

Datasets, after being scaled, are fed to Feature Selection methods such as Chi-squared, Correlation, Mutual Information, Sequential Feature Selection and proposed PFS model. The features selected by these FS methods are further input to Random Forest Classifier. Performance metrics such as Accuracy, Precision, Recall and AUC-ROC are calculated using 100-fold cross-validation, and results are analysed. Bayesian statistical validation is applied to AUC-ROC results to find out recommended Feature Selector.

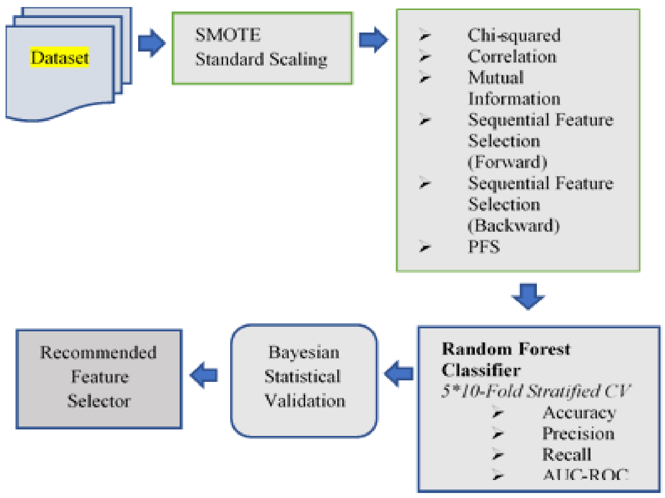


Fig. 1 Methodology

We have datasets from three publicly available Software Repositories- AEEEM [20], PROMISE [3],[21], and NASA [10]. Description of datasets is given in Table 1.

Table 1. Datasets

S. No.	Dataset	Repository	No. of Samples	No. of Features	No. of Buggy Samples	Bug %age
1	EQ	AEEEM	324	62	129	39.81
2	JDT		997	62	206	20.66
3	Lucene		691	62	64	9.26
4	MyLyn		1862	62	245	13.16
5	PDE		1497	62	209	13.96
6	kc1	NASA	2109	23	326	15.46
7	mc1		9466	40	68	0.72
8	pc2		5589	39	23	0.41
9	pc3		1563	40	160	10.24
10	pc4		1458	40	178	12.21
11	ant-1.7	PROMISE	745	22	166	22.28
12	data_arc		225	19	29	12.89
13	data_ivy-2.0		352	19	40	11.36
14	data_prop-6		644	19	61	9.47
15	data_redaktor		175	19	27	15.43
16	jedit-3.2		272	22	90	33.09
17	jedit-4.2		367	22	48	13.08
18	log4j-1.1		109	22	37	33.94
19	lucene-2.0		195	22	91	46.67
20	poi-2.0		314	22	37	11.78
21	synapse-1.0		157	22	16	10.19
22	synapse-1.2		256	22	86	33.59
23	velocity-1.6		229	22	78	34.06
24	xalan-2.4		723	22	110	15.21

3.1 SMOTE

SMOTE is a Synthetic Minority Oversampling Technique [22] that aims to reduce data imbalance by generating synthetic minority samples through interpolating several minority class instances within a defined neighbourhood.

3.2 Standardization and Scaling

Dataset is pre-processed by removing its mean and making unit variance (standardization). The further dataset is scaled to unit range (0,1). Sometimes Machine Learning techniques do not work as intended if the data does not resemble a normal distribution.

3.3 Chi-squared (Chi)

Chi-squared is a statistic that measures dependence between non-negative features and labels. It is given by the formula (1)

$$\chi 2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (1)$$

3.4 Mutual Information (MI)

Mutual information [23] quantifies the amount of information about the target class, which can be estimated from the feature. MI aims to decrease entropy or randomness between variables of interest. MI between two discrete variables, A and B, can be stated as formula (2).

$$I(A;\, B) = H(A) - H(A|B) \quad (2)$$

where
H(A) is A's entropy, and H(A|B) is the conditional entropy of A given B.

3.5 Correlation (Cor)

We have used Pearson Correlation in this study. It is a metric to calculate the linear association between two variables (value ranges -1 to 1). It is given by formula (3).
-1 (negative linear correlation)
+1(positive linear correlation)
0 indicates no correlation.

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \mathrm{a})(b_i - \mathrm{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \mathrm{a})^2}\sqrt{\sum_{i=1}^{n}(b_i - \mathrm{b})^2}} \quad (3)$$

where
a and b are the means of variables A and B.

3.6 Sequential Feature Selection

Sequential Feature Selection can be forward (SFS) or backward (SBS). SFS is a greedy procedure that keeps on adding features one at a time that improves the cross-validated score when trained on added features. SBS is similar to SFS, only that it starts with all the features and greedily removes them.

3.7 Select by Shuffle (PFS)

PFS is a hybrid feature selection mechanism (combination of filter and wrapper methods). PFS selects features by determining a drop in Classifier performance once the value of each feature is randomly permuted. The classifier performance will drop significantly (below the decided threshold) on permutation if the feature is significantly linked with the target label. Figure 2 shows the flowchart describing PFS. We have used an RF classifier, and 10-cross validated AUC-ROC metric inside PFS.

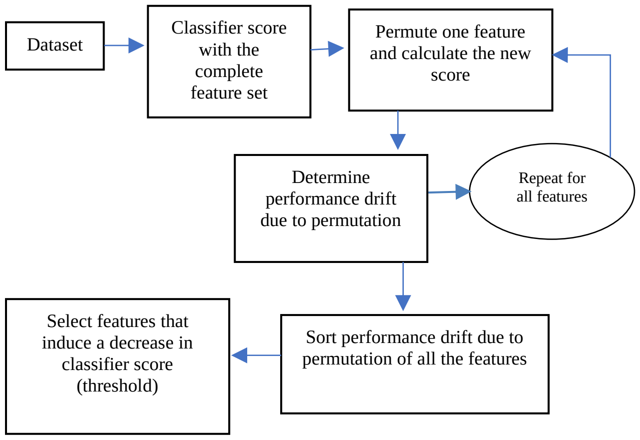

Dataset
Classifier score with the complete feature set
Permute one feature and calculate the new score
Determine performance drift due to permutation
Repeat for all features
Sort performance drift due to permutation of all the features
Select features that induce a decrease in classifier score (threshold)

Fig. 2 PFS

3.8 Performance Metrics

In this experimental study, we have collected four performance metrics – Accuracy, Precision, Recall, and Area Under Curve- Receiver Operating Characteristics (AUC-ROC). Accuracy, Precision, and Recall are calculated from the Confusion Matrix (Figure 3).

Confusion Matrix		*Classifier Prediction*	
		0	1
Actual Labels	0	True Negative (TN)	False Positive (FP)
	1	False Negative (FN)	True Positive (TP)

Fig. 3 Evaluation Matrix

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall\ or\ True\ Positive\ Rate(TPR) = \frac{TP}{TP + FN}$$

$$False\ Positive\ Rate = \frac{FP}{FP + TN}$$

ROC [25] is a graph plotting TPR vs FPR at different classification thresholds. AUC is the area under the ROC curve and is a valuable metric in comparing two classifiers [26].

3.9 Bayesian Statistical Analysis

In Bayesian statistics, “nonparametric methods typically learn distributions on function spaces, and thus effectively involve infinitely many parameters” [19]. Beneficial for smaller datasets also because prior distributions are used for handling complexity (more information make richer posteriors). We have used Bayesian Signed Rank Test [13] [14] in this experimental study to validate our results statistically. Bayesian statistical analysis has many advantages over pitfalls [17] of conventional frequentist Null Hypothesis Significance Testing (NHST)

- NHST only rejects or fails to reject the Null Hypothesis but does not provide the probability or evidence that two classifiers are equal or different [14].
- The null hypothesis is practically irrelevant because no two classifiers are perfectly equivalent. Hence, by considering many data points, significance can always be established even when the effect size is trivial.
- NHST does not provide any information regarding effect size or its uncertainty.
- Nothing practical can be established once the Null Hypothesis is not rejected.
- There is no significant or scientific cause for deciding alpha as 0.05. Entirely different analyses can be reached while considering a different alpha.

American Statistical Association has also made the case against p-values and related NHST [16]. Further, NHST is slowly losing significance in other fields of science [17].
Bayesian hypothesis testing is a promising field and a credible alternative to NHST. It returns posterior probabilities of the null and alternate hypothesis, which can be fully explained in terms of mean, variance, credible interval, and density function. Based on posterior probability, Bayesian statistics can also accept the Null hypothesis.
Bayesian Signed Rank Test [13] can be seen vis-a-vis frequentist Wilcoxon Signed Rank Test. It is based on Dirichlet Process (DP) [18]. DP is a stochastic distribution

bound by a set of probability distributions. It is used to describe prior knowledge about the distribution of random variables, i.e. modelling of data generating process.

Let y be a scalar variable, and y = {y_1,.........y_q} be a vector of independent and identically distributed observation y. We assume a DP before the probability distribution of y. DP is defined by prior strength s, which is finite and positive and prior mean that is a probability distribution G_0 on y. If G_0 = δ_{z0}, i.e. a Dirac's delta centred on pseudo observation z_0, the posterior probability function of Y can be defined as:

$$p(y) = \delta_{y_0}(y)w_0 + \sum_{j=1}^{n} \delta_y(y)w_j$$

where,

(w_0,w_1,w_2,.....w_n) ~ Dir (s,1,...,1)

The above probability function is a mixture of Dirac's deltas centred on observations y, whose weights are Dirichlet distributed with parameters (s, 1,…,1).In Bayesian Signed Rank Test, we find three probabilities, i.e., $P_{left,}$ $P_{rope,}$ $P_{right.}$ P_{rope} (ROPE – Region of Practical Equivalence) is the probability that the two classifiers are equivalent in performance. P_{left} and P_{right} are probabilities whether Classifier 1 or Classifier 2 (here, Feature Selector) is better. Figure 4 shows two cases from our experiments graphically. The figure on the left shows that the probability of *SBS* outperforming *Correlation* is high, whereas the figure on the right shows that the probability of SBS and *SFS* being significantly different is low, i.e. P_{rope} is large.

4. Results

In this experiment, we have employed 24 datasets from three different Software Repositories (AEEEM, NASA, PROMISE) and applied seven Feature Selection Techniques to reduce their dimensionality. After dimensionality reduction, we have calculated performance metrics based on Random Forest Classifier. Further, results are 10-Fold Stratified cross-validated, which is performed five times. Performance metrics include scalar metrics such as Accuracy, Precision, Recall, ROC-AUC, and Matthews Coefficient (MCC). Bayesian Statistical analysis (Bayesian Signed Rank Test) is done to arrive at the best Feature Selection technique using ROC-AUC scores.

4.1 RQ1: How have FS Techniques Performed?

We have employed three types of Feature Selection Techniques in this experimental study – Filter (Correlation, Chi-squared and Mutual Information), Wrapper (SFS and SBS), and Hybrid (Select by Shuffling). FS techniques aim to extract the most relevant dataset features given the labels. Table 2 provides values of cross-validated Accuracy for twenty-four datasets after applying RF Classifier using features selected by the above-referred FS techniques.

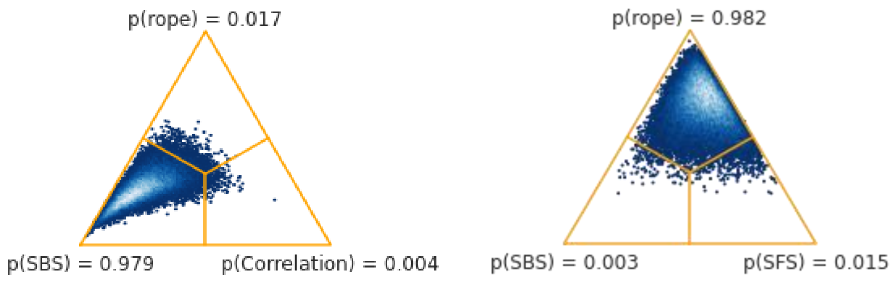


Fig. 4 Region of Practical Equivalence (ROPE)

Table 2. Accuracy

Dataset	No FS	PFS	MI	Cor	Chi2	SBS	SFS
ant-1.7	0.8016	0.8056	0.7944	0.7969	0.809	0.8046	0.813
data_arc	0.808	0.8067	0.808	0.7858	0.8031	0.82	0.8111
data_ivy-2.0	0.8486	0.8318	0.8431	0.8151	0.8418	0.8278	0.8531
data_prop-6	0.8486	0.8385	0.8454	0.838	0.8418	0.8463	0.8454
data_redaktor	0.8611	0.8514	0.8617	0.8486	0.8629	0.8731	0.848
EQ	0.7747	0.7907	0.7746	0.7713	0.7432	0.7849	0.7923
JDT	0.8551	0.847	0.8301	0.8261	0.8433	0.8544	0.8581
jedit-3.2	0.7747	0.7647	0.7691	0.7703	0.7714	0.7943	0.7916
jedit-4.2	0.8439	0.8297	0.8226	0.8216	0.8425	0.8562	0.8561
kc1	0.8321	0.8176	0.8126	0.8162	0.8088	0.8457	0.8322
log4j-1.1	0.7641	0.7609	0.7604	0.7403	0.7607	0.7791	0.8212
lucene-2.0	0.6759	0.7082	0.6733	0.6738	0.6677	0.7092	0.6769
Lucene	0.8878	0.8806	0.8747	0.8637	0.8774	0.89	0.8873
mc1	0.991	0.9902	0.9899	0.9899	0.9902	0.9897	0.9891
MyLyn	0.8608	0.8592	0.8552	0.8576	0.862	0.866	0.8735
pc2	0.9921	0.9834	0.9889	0.9887	0.9833	0.9901	0.9923
pc3	0.8592	0.8401	0.8402	0.8492	0.847	0.8758	0.8726
pc4	0.8985	0.8946	0.8844	0.8217	0.7556	0.8919	0.8984
PDE	0.8519	0.8285	0.8192	0.8086	0.8402	0.8527	0.8553
poi-2.0	0.8488	0.8232	0.8392	0.8449	0.8599	0.8548	0.8497
synapse-1.0	0.8554	0.814	0.8319	0.8183	0.8231	0.8376	0.8677
synapse-1.2	0.7629	0.7496	0.7382	0.7369	0.7477	0.7742	0.7781
velocity-1.6	0.7467	0.7128	0.7102	0.7171	0.6808	0.7462	0.7384
xalan-2.4	0.8098	0.7441	0.7351	0.7028	0.7763	0.7663	0.7103

Table 3. Precision

Dataset	No FS	PFS	MI	Cor	Chi2	SBS	SFS
ant-1.7	0.8094	0.8028	0.8026	0.7976	0.8059	0.8101	0.8133
data_arc	0.816	0.8089	0.8164	0.7893	0.7991	0.8244	0.8089
data_ivy-2.0	0.8574	0.8409	0.8369	0.8161	0.8463	0.8185	0.8446
data_prop-6	0.8509	0.8435	0.8467	0.832	0.8449	0.8517	0.8503
data_redaktor	0.8617	0.8509	0.8549	0.8549	0.8611	0.8794	0.8411
EQ	0.7797	0.7862	0.775	0.7754	0.7475	0.7784	0.8034
JDT	0.8519	0.8468	0.8334	0.8317	0.8448	0.8521	0.8524
jedit-3.2	0.7823	0.751	0.7754	0.7747	0.7739	0.7941	0.7828
jedit-4.2	0.8475	0.8294	0.8224	0.8139	0.8434	0.8651	0.8591
kc1	0.835	0.8199	0.8114	0.816	0.8094	0.8418	0.8329
log4j-1.1	0.7729	0.7552	0.7568	0.7325	0.7524	0.7745	0.831
lucene-2.0	0.6805	0.6944	0.6626	0.6564	0.6759	0.6974	0.6744
Lucene	0.8854	0.8799	0.8738	0.8608	0.8776	0.8886	0.889
mc1	0.9911	0.9903	0.9899	0.9893	0.9899	0.9901	0.9894
MyLyn	0.8626	0.8594	0.8562	0.8578	0.863	0.8656	0.8727
pc2	0.9922	0.9833	0.9886	0.9888	0.9841	0.9899	0.9923
pc3	0.8574	0.8363	0.8387	0.8466	0.8433	0.874	0.8742
pc4	0.9001	0.8938	0.8828	0.8207	0.7543	0.8933	0.8979
PDE	0.8502	0.8307	0.8185	0.811	0.84	0.8522	0.8575
poi-2.0	0.8487	0.8303	0.8472	0.8383	0.8554	0.857	0.8554
synapse-1.0	0.8477	0.8041	0.8288	0.8079	0.8338	0.841	0.8714
synapse-1.2	0.7652	0.7407	0.7345	0.7348	0.743	0.7735	0.7707
velocity-1.6	0.7371	0.7126	0.7148	0.7149	0.6739	0.7641	0.7364
xalan-2.4	0.8122	0.7453	0.7411	0.7133	0.7692	0.7754	0.748

Table 4. Recall

Dataset	No FS	PFS	MI	Cor	Chi2	SBS	SFS
ant-1.7	0.8091	0.7960	0.8013	0.7946	0.8113	0.8093	0.8142
data_arc	0.8129	0.8062	0.8218	0.7916	0.8080	0.8164	0.8080
data_ivy-2.0	0.8545	0.8281	0.8364	0.8205	0.8460	0.8270	0.8497
data_prop-6	0.8483	0.8471	0.8435	0.8388	0.8477	0.8458	0.8548
data_redaktor	0.8657	0.8594	0.8446	0.8514	0.8669	0.8646	0.8423
EQ	0.7744	0.7867	0.7682	0.7809	0.7441	0.7756	0.7970
JDT	0.8535	0.8467	0.8285	0.8286	0.8435	0.8516	0.8558
jedit-3.2	0.7787	0.7651	0.7713	0.7661	0.7769	0.7867	0.7824
jedit-4.2	0.8398	0.8269	0.8291	0.8178	0.8441	0.8578	0.8548
kc1	0.8347	0.8195	0.8129	0.8166	0.8099	0.8416	0.8300
log4j-1.1	0.7603	0.7638	0.7616	0.7432	0.7559	0.7751	0.8158
lucene-2.0	0.6692	0.7077	0.6600	0.6544	0.6677	0.7026	0.6846
Lucene	0.8899	0.8755	0.8764	0.8651	0.8787	0.8902	0.8915
mc1	0.9913	0.9905	0.9897	0.9895	0.9901	0.9900	0.9898
MyLyn	0.8632	0.8605	0.8549	0.8554	0.8642	0.8643	0.8727
pc2	0.9924	0.9831	0.9890	0.9884	0.9833	0.9903	0.9920
pc3	0.8575	0.8367	0.8376	0.8461	0.8429	0.8732	0.8739
pc4	0.8986	0.8892	0.8846	0.8222	0.7550	0.8933	0.8964
PDE	0.8488	0.8310	0.8174	0.8096	0.8386	0.8504	0.8554
poi-2.0	0.8544	0.8197	0.8411	0.8431	0.8586	0.8538	0.8516
synapse-1.0	0.8537	0.8104	0.8435	0.8127	0.8284	0.8422	0.8701
synapse-1.2	0.7684	0.7344	0.7446	0.7488	0.7469	0.7790	0.7738
velocity-1.6	0.7476	0.7088	0.7076	0.7161	0.6763	0.7543	0.7345
xalan-2.4	0.8158	0.7440	0.7352	0.7118	0.7698	0.7682	0.7295

Values in bold are maximum values for that dataset, whereas italicized values are minimum values. Underlined values are values greater than No FS values for that particular dataset. Considering Table 2 - Accuracy, Table 3 - Precision, and Table 4 – Recall, SFS, and SBS FS techniques have outperformed other FS techniques in terms of Accuracy.

Further, except few datasets, FS techniques have outperformed No FS.

Table 5 represents the number of features selected by FS Techniques vis-a-vis the original number of features. We have allowed PFS to choose features and then forced other FS techniques to choose the same number of features for a proper comparison. Average Dimensionality reduction comes out to be 62.56 % for all the twenty-four datasets.

Table 5. Dimensionality reduction

Dataset	No. of Features		Dimensionality Reduction %
	No FS	FS Techniques	
ant-1.7	20	8	60.00
data_arc	17	9	47.06
data_ivy-2.0	17	7	58.82
data_prop-6	17	5	70.59
data_redaktor	17	8	52.94
EQ	60	22	63.33
JDT	60	23	61.67
jedit-3.2	20	9	55.00
jedit-4.2	20	8	60.00
kc1	21	8	61.90
log4j-1.1	20	7	65.00
lucene-2.0	20	10	50.00
Lucene	60	25	58.33
mc1	38	22	42.11
MyLyn	60	24	60.00
pc2	37	11	70.27
pc3	38	14	63.16
pc4	38	4	89.47
PDE	60	23	61.67
poi-2.0	20	8	60.00
synapse-1.0	20	8	60.00
synapse-1.2	20	8	60.00
velocity-1.6	20	4	80.00
xalan-2.4	20	2	90.00

While selecting a Machine Learning method, it is important to observe the computational power required to process it. We have taken the time (in seconds) as a proxy to the computational load of an FS Technique. Figure 5 shows a comparative graph of the time taken by all the twenty-four datasets in processing.

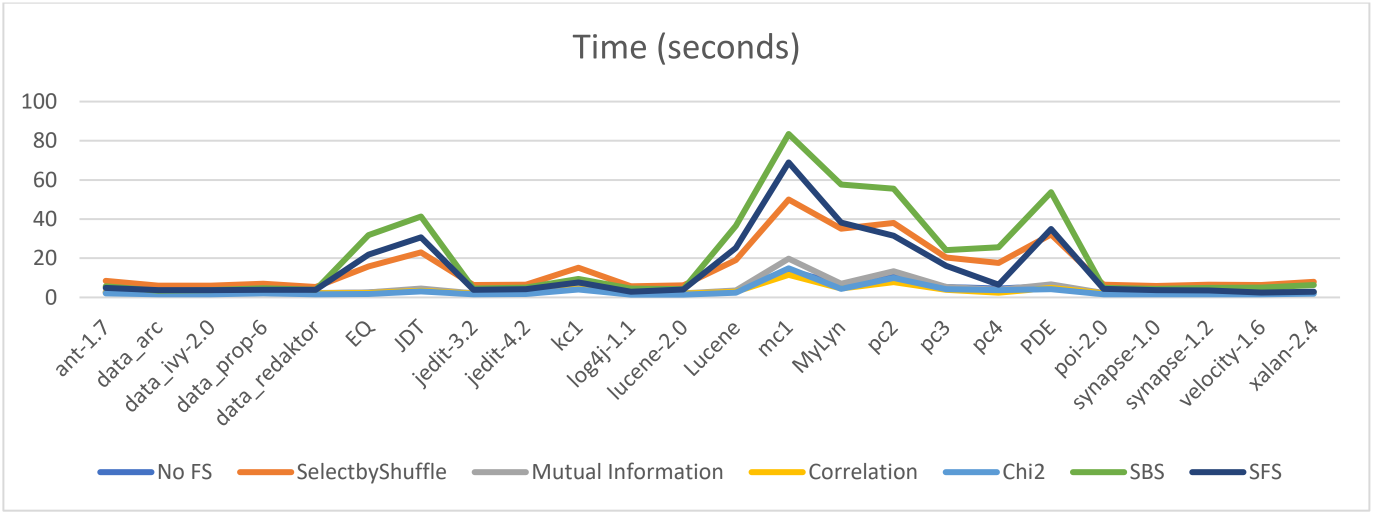


Fig. 5 Processing time

It can be seen from Figure 5 SBS, SFS, and PFS have taken maximum time to compute the selected cross-validated performance metrics. Further, as either the number of features or samples increases, computation time increases across all considered techniques. Table 6 depicts AUC-ROC scores of FS Technique-based RF Classifier.

It can be observed that No FS has outperformed in the maximum number of datasets.

Wrapper-based methods (SBS and SFS) have, in general, performed better than Hybrid method (Select by Shuffle) filter-based methods (Chi-squared, Correlation, Mutual Information).

Table 6. AUC-ROC

Dataset	No FS	PFS	MI	Cor	Chi2	SBS	SFS
ant-1.7	0.8257	0.8168	0.8192	0.8125	0.7726	0.7761	0.8157
data_arc	0.6408	0.6532	0.6530	0.6332	0.6199	0.6326	0.6060
data_ivy-2.0	0.7861	0.7743	0.7701	0.7538	0.7838	0.7055	0.7374
data_prop-6	0.7460	0.7320	0.6779	0.6528	0.7099	0.6496	0.6897
data_redaktor	0.7777	0.7998	0.7980	0.7693	0.7831	0.7977	0.8049
EQ	0.8399	0.8489	0.8408	0.8378	0.8112	0.8423	0.8704
JDT	0.8714	0.8723	0.8268	0.8225	0.8408	0.8659	0.8566
jedit-3.2	0.8346	0.8145	0.8052	0.8269	0.8245	0.8244	0.8295
jedit-4.2	0.8283	0.8123	0.8105	0.8064	0.8286	0.8392	0.8183
kc1	0.6895	0.6853	0.6832	0.6386	0.7512	0.7409	0.7053
log4j-1.1	0.8151	0.8285	0.8013	0.7994	0.7915	0.7834	0.8513
lucene-2.0	0.7463	0.7644	0.7422	0.7313	0.7286	0.7651	0.7459
Lucene	0.8035	0.8105	0.7634	0.7375	0.7391	0.7987	0.7973
mc1	0.9428	0.9261	0.9496	0.9542	0.9441	0.9517	0.9549
MyLyn	0.8114	0.8129	0.7866	0.7489	0.7853	0.7955	0.8053
pc2	0.6702	0.6343	0.6223	0.5518	0.7022	0.6577	0.6795
pc3	0.8126	0.8193	0.8074	0.8178	0.8199	0.7980	0.8274
pc4	0.9300	0.9151	0.9084	0.8826	0.6745	0.9180	0.9106
PDE	0.7785	0.7622	0.7475	0.7452	0.7530	0.7553	0.7767
poi-2.0	0.8087	0.7459	0.7870	0.7807	0.7379	0.7636	0.7949
synapse-1.0	0.7280	0.6921	0.7588	0.6950	0.7366	0.6935	0.6952
synapse-1.2	0.8014	0.7947	0.7862	0.7798	0.7849	0.8048	0.8020
velocity-1.6	0.7912	0.7395	0.7431	0.7274	0.7151	0.7692	0.7528
xalan-2.4	0.7858	0.7533	0.7235	0.7002	0.7521	0.7246	0.5175

4.2 RQ2: What insights can be drawn from Bayesian Statistics?

This experimental study has undertaken a Bayesian statistical analysis of AUC-ROC results. We performed Shapiro-Wilk Test along with Bonferroni Correction on AUC-ROC Scores (7 populations with 24 paired samples) and found populations to be normal. Table 7 lists Mean, Standard Deviation, Confidence Interval-Lower, and Upper and Cohen's Delta (Cohen's d) according to Bayesian Signed Rank Test. Effect Size is calculated compared to the maximum mean value (here, NO FS) using Cohen's d.

In Bayesian Signed Rank Test, the comparison is performed according to Posterior probabilities values. Region of Practical Equivalence (ROPE) has been set dynamically as 0.1 * Cohen'

Table 7. Bayesian analysis

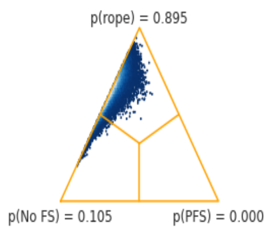
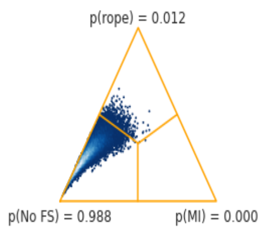
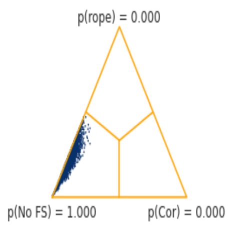
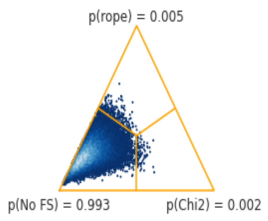
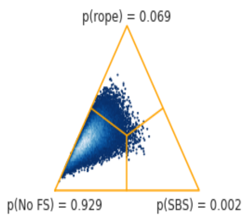
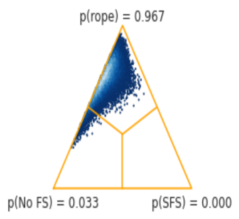
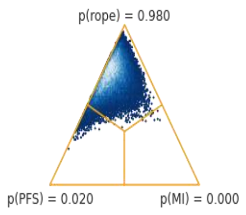
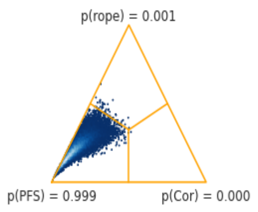
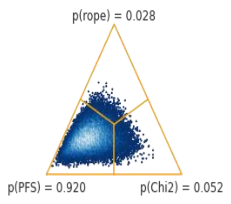
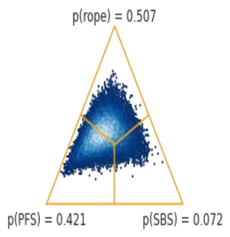
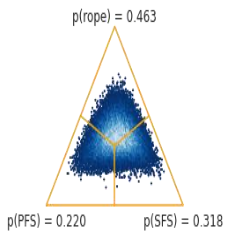
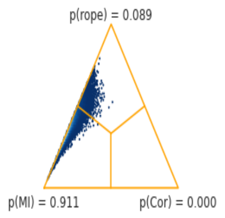
Technique	Mean	Standard Deviation	Confidence Interval		Cohen's d	Effect Size
			Lower	Upper		
No FS	0.7944	0.0702	0.7405	0.8483	0.0000	Negligible
Select by Shuffle	0.7837	0.0732	0.7275	0.8398	0.1497	Negligible
SBS	0.7772	0.0779	0.7175	0.8370	0.2318	Small
SFS	0.7769	0.0956	0.7035	0.8502	0.2090	Small
Mutual Information	0.7755	0.0734	0.7191	0.8319	0.2631	Small
Chi2	0.7663	0.0644	0.7168	0.8157	0.4175	Small
Correlation	0.7586	0.0870	0.6918	0.8253	0.4533	Small

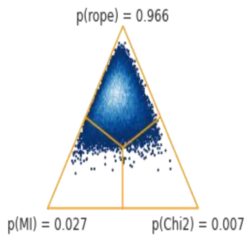
Table 8 lists pair-wise Posterior Probabilities. (*,*,*) should be read as (p_smaller, p_equal, p_large). In Table 8, column-2, row -1, the probability that Select by Shuffle is smaller than NO FS is 0.8575, the probability that Select by Shuffle is equal to NO FS is 0.1466, and the probability that Select by Shuffle is larger than NO FS is 0.00084. Table 9 shows graphically how posterior probability varies between FS technique pairs and ROPE intervals.

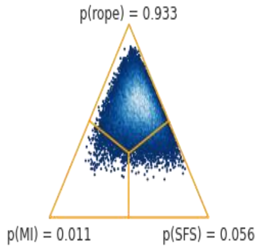
Table 8. Posterior prabability

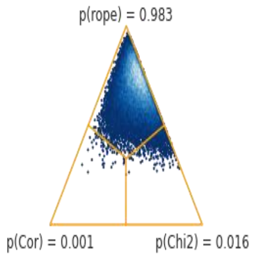
FS METHOD		No FS	PFS	SBS	SFS	MI	Chi2	Corr.
No FS	p_smaller		0.8575	0.98278	0.76124,	0.99878	0.99894	0.99998
	p_equal		0.14166	0.01378	0.21028,	0.00102	8e-05	2e-05
	p_large	----	0.00084	0.00344	0.02848	0.0002	0.00098	0.0
PFS	p_smaller			0.71872	0.2701	0.88704	0.95334	0.9998
	p_equal			0.1016	0.30622	0.08288	0.00018	0.0001
	p_large	----	----	0.17968	0.42368	0.03008	0.04648	0.0001
SBS	p_smaller				0.1051	0.6795	0.77294	0.98846
	p_equal				0.16894	0.01578	0.00404	0.00246
	p_large	----	----	----	0.72596	0.30472	0.22302	0.00908
SFS	p_smaller					0.92326	0.9047	0.9969
	p_equal					0.00472	0.00048	0.00128
	p_large	----	----	----	----	0.07202	0.09482	0.00182
MI	p_smaller						0.7322	0.9701
	p_equal						0.0091	0.0299
	p_large	----	----	----	----	----	0.2587	0.0
Chi2	p_smaller							0.84812
	p_equal							0.0351
	p_large	----	----	----	----	----	----	0.11678
Corr		----	----	----	----	----	----	----

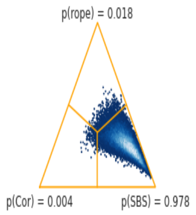
Table 9. Graphical Representations of Posterior Probability using ROPE=0.01

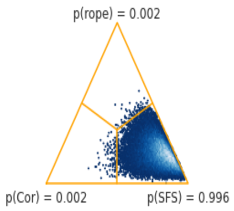
		
		
		
		

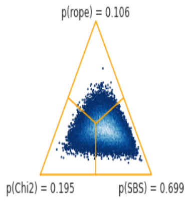


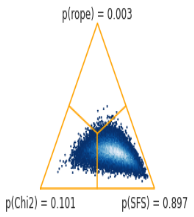


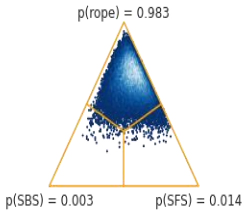












Considering posterior probabilities calculated in Table 8 and significance level as 0.05, the following can be inferred:

- The mean value of the population No FS is larger th an that of the population's SBS, Mutual Information, Chi2, and Correlation.
- The mean value of the population Select by Shuffle is larger than that of the population's Chi2 and Corre lation.
- The mean value of the population SBS is larger than that of the population's Correlation.
- The mean value of the population SFS is larger than that of the population's Correlation.
- The mean value of the population Mutual Informati on is larger than that of the population's Correlation.
- All other differences are inconclusive.

4.3 RQ3: Which FS technique is best amongst the considering techniques?

Based on the aforementioned Bayesian analysis, there is no conclusive difference between *No FS*, *SFS*, and *PFS*. Hence, based on the number of features selected, SFS and

PFS outperform in dimensionality reduction. Further, *SFS* is more computationally expensive than *PFS* (Figure II). Therefore, it can be deduced that *PFS* is the best performing FS technique amongst other considered techniques.

5. Conclusion

We examined six different FS techniques viz. PFS, Mutual Information, Correlation, Chi-squared, Sequential Forward, and Backward Feature Selection on twenty-four datasets using Random Forest Classifier. Bayesian Signed Rank Test was performed on AUC-ROC scores of all FS Techniques, and it was found that SFS, PFS has performed equivalent to model without any FS. PFS outperformed SFS in terms of lower computational time. We reduced the curse of dimensionality of twenty-four datasets by 62.56 % and maintained the original classifier performance.
We recommend our model Permutation-based Hybrid Feature Selection Model (PFS) in any Machine Learning model to reduce dimensionality and computational cost. Future work in this direction can examine the effect of changing the classifier and metric used in PFS.

Declarations

Ethics approval and consent to participate

The authors declare that we conform to Ethical Responsibilities as referred to in Submission Guidelines.

Consent for publication

The authors give consent to Journal to publish this original work.

Availability of data and material

The dataset used in this study is from three repositories- AEEEM, PROMISE and NASA, which are publicly available free.

Competing Interests

The authors did not receive support from any organization for the submitted work.

Funding

The authors have no relevant financial or non-financial interests to disclose.

Authors' Contributions

TS performed the experimental setup under the guidance of OPS. Both authors read and approved the final manuscript.

References

[1] Devi CA, Kannammal KE, Surendiran B, A Hybrid Feature Selection Model for Software Bug Prediction. Int. J. Comput. Sci. Appl. 2(2) (2012) 25-35.

[2] Gayatri N, Nickolas S, Reddy AV, Performance Analysis and Enhancement of Software Quality Metrics Using Decision Tree-Based Feature Extraction, International Journal of Recent Trends in Engineering. 2(4) (2009) 1-54.

[3] The PROMISE Repository of Software Engineering Databases. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[4] Khan B, Naseem R, Shah MA, Wakil K, Khan A, Uddin MI, Mahmoud M, Software Bug Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques, Journal of Healthcare Engineering. 15 (2021) 2021.

[5] Menzies T, Greenwald J, Frank A, Data Mining Static Code Attributes to Learn Bug Predictors. IEEE Transactions on Software Engineering. 33(1) (2006) 2-13.

[6] Song Q, Jia Z, Shepperd M, Ying S, Liu J, A General Software Bug-Proneness Prediction Framework. IEEE Transactions on Software Engineering. 37(3) (2010) 356-70.

[7] Agarwal S, Tomar D. A, Feature Selection-Based Model for Software Bug Prediction, Assessment. (2014) 65.

[8] Liu S, Chen X, Liu W, Chen J, Gu Q, Chen D, FECAR: A Feature Selection Framework for Software Bug Prediction, In 2014 IEEE 38th Annual Computer Software and Applications Conference, IEEE. 21 (2014) 426-435.

[9] Khoshgoftaar TM, Gao K, Napolitano A, Wald R, A Comparative Study of Iterative and Non-Iterative Feature Selection Techniques for Software Bug Prediction, Information Systems Frontiers. 16(5) (2014) 801-22.

[10] Balogun AO, Basri S, Abdulkadir SJ, Hashim AS. Performance Analysis of Feature Selection Methods in Software Bug Prediction: A Search Method Approach, Applied Sciences. 9(13) (2019) 2764.

[11] Catal, Cagatay, and Banu Diri, Investigating the Effect of Dataset Size, Metrics Sets, and Feature Selection Techniques on Software Fault Prediction Problem, Information Sciences. 179(8) (2009) 1040-1058.

[12] Jakhar, Amit Kumar, and Kumar Rajnish, Software Fault Prediction with Data Mining Techniques by Using Feature Selection Based Models, International Journal on Electrical Engineering & Informatics. 10(3) (2018).

[13] Benavoli, Alessio, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri, A Bayesian Wilcoxon Signed-Rank Test Based on the Dirichlet Process, in International Conference on Machine Learning, PMLR. (2014) 1026-1034.

[14] Benavoli, Alessio, Giorgio Corani, Janez Demšar, and Marco Zaffalon, Time for a Change: A Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis. The Journal of Machine Learning Research. 18(1) (2017) 2653-2688.

[15] Raftery, Adrian E. Bayesian, Model Selection in Social Research, Sociological Methodology. (1995) 111-163.

[16] Wasserstein, Ronald L, and Nicole A. Lazar, The ASA Statement on P-Values: Context, Process, and Purpose. (2016) 129-133.

[17] Trafimow, David, Valentin Amrhein, Corson N. Areshenkoff, Carlos J. Barrera-Causil, Eric J. Beh, Yusuf K. Bilgiç, Roser Bono et al., Manipulating the Alpha Level Cannot Cure Significance Testing, Frontiers in Psychology. 9 (2018) 699.

[18] Ferguson, Thomas S, A Bayesian Analysis of Some Nonparametric Problems, the Annals of Statistics. (1973) 209-230.

[19] Bernardo, José M, and Adrian FM Smith, Bayesian theory, John Wiley & Sons. 405 (2009).

[20] D'Ambros, Marco, Michele Lanza, and Romain Robbes. An Extensive Comparison of Bug Prediction Approaches. In 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE. (2010) 31-41.

[21] Shirabad, J. Sayyad, and T. J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa. (2005).

[22] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-Sampling Technique, Journal of Artificial Intelligence Research. 16 (2002) 321-357.

[23] Kraskov, Alexander, Harald Stögbauer, and Peter Grassberger, Estimating Mutual Information, Physical Review. 69(6) (2004) 066138.

[24] Ferri, Francesc J, Pavel Pudil, Mohamad Hatef, and Josef Kittler, Comparative Study of Techniques for Large-Scale Feature Selection, in Machine Intelligence and Pattern Recognition, North-Holland. 16 (1994) 403-413.

[25] Metz, Charles E. Basic Principles of ROC Analysis. In Seminars in Nuclear Medicine, WB Saunders. 8(4) (1978) 283-298.

[26] Fawcett, Tom, An introduction to ROC Analysis, Pattern Recognition Letters. 27(8) (2006) 861-874.

[27] Breiman, Leo, Random Forests, Machine Learning. 45(1) (2001) 5-32.

[28] Breiman, Leo, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone, Classification and Regression Trees, Routledge. (2017).

[29] Fenton, Norman E, and Martin Neil, A Critique of Software Defect Prediction Models, IEEE Transactions on Software Engineering. 25(5) (1999) 675-689.

[30] Menzies, Tim, Zach Milton, Burak Turhan, Bojan Cukic, Yue Jiang, and Ayşe Bener, Defect Prediction from Static Code Features Current Results, Limitations, New Approaches, Automated Software Engineering. 17(4) (2010) 375-407.

[31] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al., Scikit-Learn: Machine Learning in Python, The Journal of Machine Learning Research. 12 (2011) 2825-2830.

[32] Herbold, Steffen, Autorank: A Python Package for Automated Ranking of Classifiers, Journal of Open Source Software. 5(48) (2020) 2173.