Feature Selection in Cross-Project Software Defect Prediction

Feature Selection in Cross-Project Software Defect Prediction

A Saifudin[1,2*], A Trisetyarso[2], W Suparta[3], C H Kang[4], B S Abbas[2], Y Heryadi[2]
[1]Informatics Engineering, Pamulang University, Jalan Raya Puspitek 46, Banten 15310, Indonesia
[2]Doctor of Computer Science Program, Bina Nusantara University, Jalan Kebon Jeruk Raya 27, Jakarta 11530, Indonesia
[3]Informatics Department, Pembangunan Jaya University, Banten, Indonesia
[4]Department of Electronics and Communication Engineering, Kwangwoon University, South Korea

[*]aries.saifudin@unpam.ac.id

Abstract. Advances in technology have increased the use and complexity of software. The complexity of the software can increase the possibility of defects. Defective software can cause high losses. Fixing defective software requires a high cost because it can spend up 50% of the project schedule. Most software developers don't document their work properly so that making it difficult to analyse software development history data. Software metrics which use in cross-project software defects prediction have many features. Software metrics usually consist of various measurement techniques, so there are possibilities for their features to be similar. It is possible that these features are similar or irrelevant so that they can cause a decrease in the performance of classifiers. In this study, several feature selection techniques were proposed to select the relevant features. The classification algorithm used is Naive Bayes. Based on the analysis using ANOVA, the SBS and SBFS models can significantly improve the performance of the Naïve Bayes model.

1. Introduction

The use of software has increased with the development of technology. Software that provides greater benefits usually has high complexity. Software complexity is directly proportional to the defects contained in it [1]. A software defect is a bug that causes the software which develop can't meet expectation[2] or error, fault, flaw, or failure in the software that causes system produces an unexpected or incorrect outcome[3]. Software defects can cause large losses if not corrected immediately.

To find and correct software defects are generally done by testing. Testing takes a lot of time and costs compared to other stages in software development [4]. So, we need a method that can be used to estimate the location of software defects in order to find defects faster with lower costs.

To estimate the location of software defects can be done by analyzing software metrics from past projects using machine learning. There are not many developers who collect software development history. If we don't have enough local data, we can use datasets from other project [5]. The use of limited historical data for software defect prediction has attracted the attention of researchers and practitioners[6]. Software defect prediction techniques use datasets from other different projects known as cross-software project defects predictions [7].

Generally, the software metrics used to predict cross-project software defects have many features. Software metrics usually consist of various measurement techniques, so there are possibilities for their features to be similar. The features collected also have the possibility of being irrelevant to predict software defects so that it can cause a decrease in the performance of classifiers[8].

This research proposes to implement feature selection to select relevant features. On the feature selection, the algorithm will choose the feature which gives a high reward to the model performance[9]. Several feature selection techniques were proposed are Sequential Forward Selection (SFS), Sequential Backward Selection (SBS)[10], Sequential Forward Floating Selection (SFFS), Sequential Backward Floating Selection (SBFS)[11], and SelectKBest which will select k number of feature with highest scores[12]. The classification algorithm which use to classify is Naive Bayes.

2. Methods

This experiment carried out by proposing software defect prediction models, then applying to software metrics dataset. The results of model performance measurements are compared to get the best model.

The proposed model implements using NASA dataset because it is the most widely used dataset in this study so that it is easy to compare with other researchers. The NASA dataset is obtained from https://github.com/klainfo/NASADefectDataset which is a backup of http://nasa-softwaredefectdatasets.wikispaces.com/ from Shepperd et al. (2014). NASA datasets consist of 10 datasets, but for this work, we use five datasets which have the same attributes, namely CM1, MW1, PC1, PC3, and PC4.

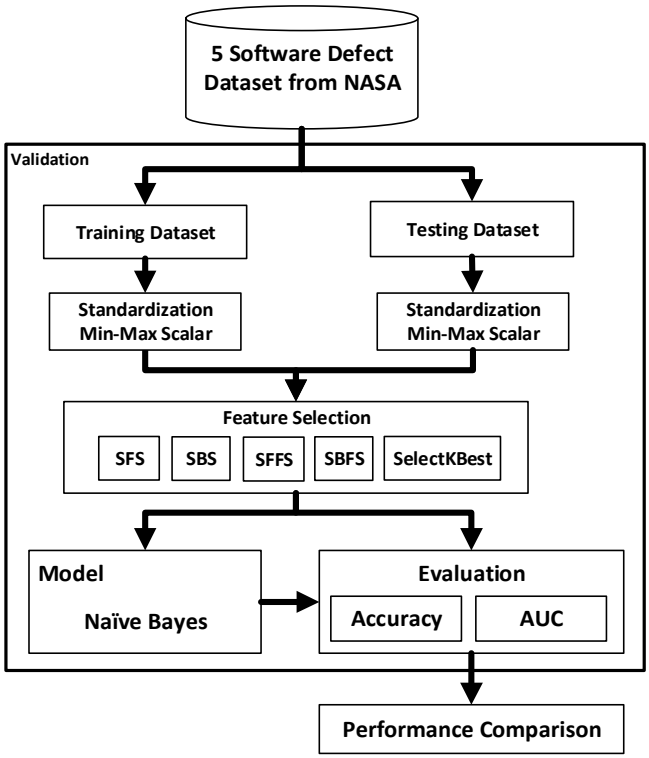

Figure 1. Proposed Model

Feature selection algorithms which have proposed is implemented to select the relevant features for the classifier. The proposed model is shown in Figure 1. Software metrics datasets that have been collected divide into two groups, one as testing dataset and the others training dataset. Then applied to standardization using min-max scalar and feature selection algorithm. The feature selection algorithm

will be analyzing the training dataset and chooses relevant features, and chooses the same features in the testing dataset.

The new dataset uses for train and tests the proposed model. This process will be repeating until all dataset has been training data. The test results are entered in the confusion matrix table and calculate the performance of classifiers is carried out in the form of accuracy and AUC (Area Under the Curve).

Based on the proposed model, there will be 5 models, namely NB, SFS, SBS, SFFS, SBFS, and KBest. The performance of the five models is compared to get the best model. SFS is a deterministic feature selection method that uses hill-climbing search to add and assess all possible single attribute expansions to the present subset[13]. While SBS works in the opposite direction to SFS[14]. SFS and SBS select features in one-way, so the features that have been evaluating cannot be selected again, but these weaknesses avoided in SFFS and SBFS[15]. SelectKBest is a module in the scikit learn library that select k feature that has the highest score. The score is calculated based on univariate statistical analysis, which is an analysis of variables one by one.

3. Results

Experiments carried out by applying the model using a dataset that has been collected. The model implementation using a dataset from NASA follows the proposed model as shown in Figure 1. The accuracy and AUC values of the resulting model are then visualized using the graph shown in Figure 2 and Figure 3. Figure 2 shows that the average model accuracy decreases as the number of features increases. While Figure 3 shows that the AUC value, in general, has increased.

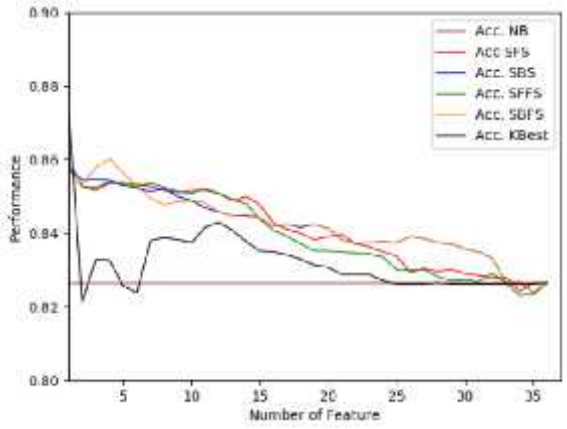

Figure 2. Graph of Accuracy models

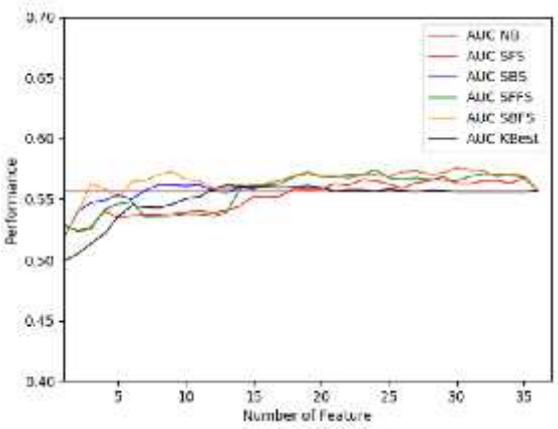

Figure 3. Graph of AUC models

To find out the best model, it is necessary to do a statistical analysis based on the performance value of the model. Statistical analysis was carried out using ANOVA (Analysis of Variance). The significance value (denoted as α or alpha) is set to 0.01. The analysis is done by calculating the p-value of the two models in pairs and turns. The resulting p-values are shown in Table 1.

The initial hypothesis (H_0) states that all models have the same mean value (H_0: $\mu_1 = \mu_2$). If the p-value is smaller than the significance value (α), it is stated to have a significant difference. Significant values (p-value) and significantly different are written in bold in Table 1.

Based on Table 1 shows that all models have significantly different values to models that do not use feature selection. The KBest accuracy value is not significantly different from SFFS but is significantly different from SFS, SBS, and SBFS.

To find out the significant difference towards better or decreasing visualization using a boxplot diagram as shown in Figure 4. Figure 4 shows that the five feature selection models can significantly increase the accuracy of Naïve Bayes classifiers.

Table 1. P-value and Significantly Different Comparison of Accuracy

	P-value Comparison						Significantly Different Comparison					
Model	**NB**	**SFS**	**SBS**	**SFFS**	**SBFS**	**KBest**	**NB**	**SFS**	**SBS**	**SFFS**	**SBFS**	**KBest**
NB	1,0000	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0007**	Not	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**
SFS	**0,0000**	1,0000	0,5958	0,6638	0,5804	**0,0024**	**Sig**	Not	Not	Not	Not	**Sig**
SBS	**0,0000**	0,5958	1,0000	0,3363	0,9806	**0,0003**	**Sig**	Not	Not	Not	Not	**Sig**
SFFS	**0,0000**	0,6638	0,3363	1,0000	0,3260	0,0106	**Sig**	Not	Not	Not	Not	Not
SBFS	**0,0000**	0,5804	0,9806	0,3260	1,0000	**0,0003**	**Sig**	Not	Not	Not	Not	**Sig**
KBest	**0,0007**	**0,0024**	**0,0003**	0,0106	**0,0003**	1,0000	**Sig**	**Sig**	**Sig**	Not	**Sig**	Not

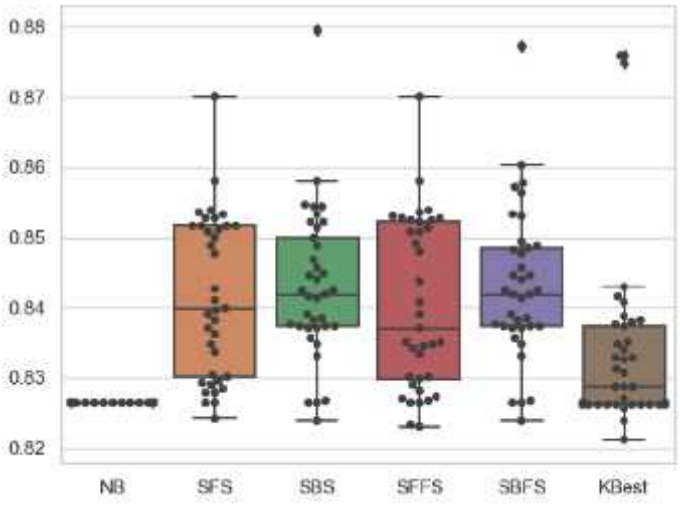

Figure 4. Boxplot visualization of Accuracy

For unbalanced data, it is recommended to measure the performance of the model based on AUC values, because it uses a balance value between True Positive Rate and True Negative Rate. The results of AUC measurements were also statistically analyzed using ANOVA. The results of the ANOVA analysis and the significance analysis are shown in Table 2.

Table 2. P-value and Significantly Different Comparison of AUC

	P-value Comparison						Significantly Different Comparison					
Model	**NB**	**SFS**	**SBS**	**SFFS**	**SBFS**	**KBest**	**NB**	**SFS**	**SBS**	**SFFS**	**SBFS**	**KBest**
NB	1,0000	**0,0059**	0,1322	0,3540	0,0117	**0,0043**	Not	**Sig**	Not	Not	Not	**Sig**
SFS	**0,0059**	1,0000	**0,0030**	0,2595	**0,0003**	0,6251	**Sig**	Not	**Sig**	Not	**Sig**	Not
SBS	0,1322	**0,0030**	1,0000	0,0930	0,4968	**0,0018**	Not	**Sig**	Not	Not	Not	**Sig**
SFFS	0,3540	0,2595	0,0930	1,0000	0,0206	0,1404	Not	Not	Not	Not	Not	Not
SBFS	0,0117	**0,0003**	0,4968	0,0206	1,0000	**0,0002**	Not	**Sig**	Not	Not	Not	**Sig**
KBest	**0,0043**	0,6251	**0,0018**	0,1404	**0,0002**	1,0000	**Sig**	Not	**Sig**	Not	**Sig**	Not

Based on Table 2 shows that there are only 2 models that show a significant difference to the Naïve Bayes model. SFFS has no difference with other models.

To show the difference significantly towards better or decreasing visualization using boxplot diagram as shown in Figure 5. Based on the visualization in Figure 5 shows that the SBS and SBFS models have significantly better differences than the Naïve Bayes model without feature selection.

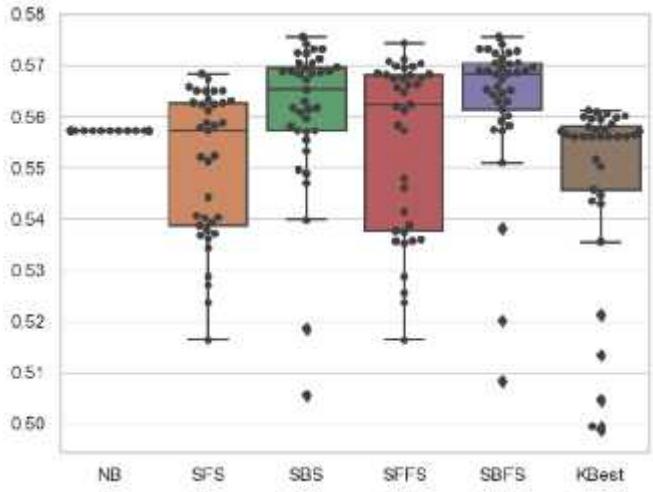

Figure 5. Boxplot visualization of AUC

4. Conclusion

The experimental results show that feature selection can improve the accuracy of the model. Based on statistical analysis using ANOVA on the value of Accuracy and AUC the feature selection model that has been applied can be concluded that the SBS and SBFS models can significantly improve the performance of the Naïve Bayes model on software defect predictions.

References

[1] Adak M F 2018 Software defect detection by using data mining based fuzzy logic *2018 Sixth Int. Conf. Digit. Information, Networking, Wirel. Commun.* 65–9
[2] Malhotra R and Kamal S 2017 Tool to handle imbalancing problem in software defect prediction using oversampling methods *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017* **2017-Janua** 906–12
[3] Prasad M C M, Florence L and Arya A 2015 A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques **8** 179–90
[4] Zhang Y, Lo D, Xia X and Sun J 2018 Combined Classifier for Cross-project Defect Prediction: An Extended Empirical Study *Front. Comput. Sci.* **12** 280–96
[5] Ryu D and Baik J 2016 Effective multi-objective naïve Bayes learning for cross-project defect prediction *Appl. Soft Comput. J.* **49** 1062–77
[6] Zhang F, Zheng Q, Zou Y and Hassan A E 2016 Cross-project defect prediction using a connectivity-based unsupervised classifier *2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Cross-project* pp 309–20
[7] Yu Q, Jiang S and Zhang Y 2017 A Feature Matching and Transfer Approach for Cross-Company Defect Prediction *J. Syst. Softw.* **132** 366–78
[8] Turabieh H, Mafarja M and Li X 2019 Iterated feature selection algorithms with layered recurrent neural network for software fault prediction *Expert Syst. Appl.* **122** 27–42
[9] Chaudhry M U and Lee J H 2018 MOTiFS: Monte Carlo Tree Search based feature selection *Entropy* **20** 1–16
[10] Paul S and Das S 2015 Simultaneous feature selection and weighting - An evolutionary multi-

objective optimization approach *Pattern Recognit. Lett.* **65** 51–9
[11] Homsapaya K and Sornil O 2018 Modified Floating Search Feature Selection Based on Genetic Algorithm *MATEC Web Conf.* **164** 01023
[12] Nair R and Bhagat A 2019 Feature selection method to improve the accuracy of classification algorithm *Int. J. Innov. Technol. Explor. Eng.* **8** 124–7
[13] Hira Z M and Gillies D F 2015 A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data *Adv. Bioinformatics* **2015** 1–13
[14] Liu H, Jiang H and Zheng R 2016 The Hybrid Feature Selection Algorithm Based on Maximum Minimum Backward Selection Search Strategy for Liver Tissue Pathological Image Classification *Comput. Math. Methods Med.* **2016**
[15] Xue B, Zhang M, Browne W N and Yao X 2016 A Survey on Evolutionary Computation Approaches to Feature Selection *IEEE Trans. Evol. Comput.* **20** 606–26