Dimensional Reduction on Cross Project Defect Prediction

A Saifudin[1*], Y Yulianti[1]
Informatics Engineering, Pamulang University, Jalan Raya Puspitek 46, Banten 15310, Indonesia

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

Abstract. The complexity of the software can increase the possibility of defects. Defective software can cause high losses. The software containing defects can cause large losses. Most software developers don't document their work properly so that making it difficult to analyse software development history data. The cross-project software defect prediction used several datasets from different projects and combining for training and testing. The dataset with high dimension can cause bias, contain irrelevance data, and require large resources to process it. In this study, several dimensional reduction algorithm and Decision Tree as classifier. Based on the analysis using ANOVA, all models that implement dimensional reduction can significantly improve the performance of the Decision Tree model.

1. Introduction

A software defect is fault or bug in the software that causes error and system produces an unexpected or incorrect outcome [1], so the system can't meet expectation [2]. The software containing defects can cause large losses. To find and repair defects requires additional costs for development. Defective software can also cause great damage or financial loss [3]. For example, software used to control drones, if they contain defects, can cause crashes. Software for banking transactions or sales that contain defects will cause financial losses.

The software has been developed increasingly complex to increase benefits, as well as security. Increased software complexity causes proportional defects to increase [4]. To guarantee that the software is errors free, generally done by testing, even though testing is the most expensive stage of software development. Efforts to reduce errors and ensure software quality is a difficult challenge [5].

Prediction of software defects is one of the efforts made to improve the quality of software and reduce development costs [6]. Software defect prediction is used to predict program units that tend to contain defects so that it can help developers to test properly and find defects quickly.

To estimate the tendency of defects in software generally, use quality metrics or static codes metrics [4]. Much metric software has been used, including Software Change Matrices (SCM) and Code Based Matrices (CBM)[7].

Many software developers ignore the importance of project documentation, so they don't have enough data to use in predicting software defects. Many researchers and practitioners are challenged to utilize limited historical data to predict software defects [8]. Software defect prediction using historical data from another project is called cross-project defect prediction [9].

The cross-project software defect prediction used several datasets from different projects. The datasets are combined to train the model and be tested using a dataset from other projects. By combining dataset from several projects, the dataset has a large dimension. The dataset with high dimension can cause bias, contain irrelevance data, and require large resources to process it. So, in this study, proposed to implement the dimension reduction algorithm, namely PCA (Principal Component Analysis), Kernel PCA, Sparse PCA, Truncated SVD, Incremental PCA, Gaussian Random Projection, dan Sparse Random Projection.

2. Methodology

NASA's dataset is used in this study because it is common for this topic. 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 contain 10 datasets, but for this work, we use datasets which have the same attributes, namely CM1, MW1, PC1, PC3, and PC4. The datasets have been processed based on the initial processing algorithm proposed by Shepperd, Song, Sun, and Mair[10] to eliminate implausible value, inconsistent data, and conflicting feature value.

The proposed Cross-project Defect Prediction Model Framework shown in Figure 1. The dataset is divided into two-part for train models and test models. Both datasets are standardized using min-max scalar and applied to the dimensional reduction algorithm. The reduced dataset is used to train and test the model. Model evaluation is done by comparing the performance of the model to get the best model. Decision Tree algorithm use as a classification algorithm because quite reliable and efficient in generating a classifier model[11].

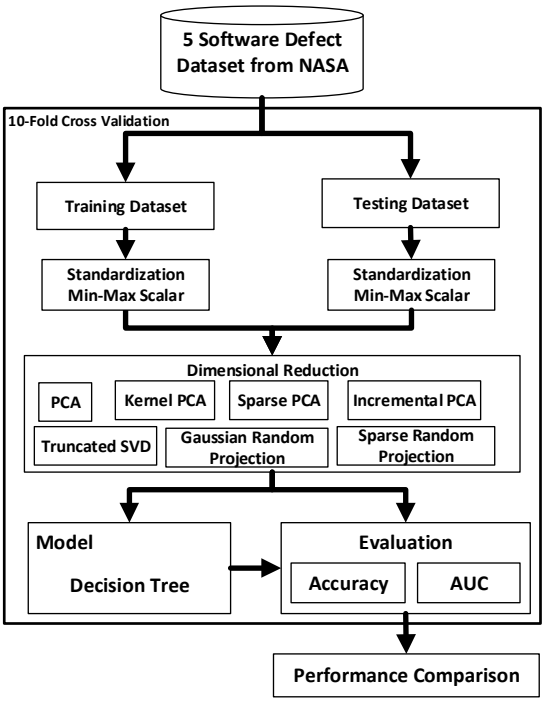

Figure 1. Cross-project Defect Prediction Model Framework

The purpose of this model is to predict defect prone modules in other projects. The proposed model is applied using 5 datasets from NASA. These datasets will be chosen alternately as testing data and the other as training data until all datasets have been testing data. The distribution of the dataset as training data and testing data is shown in Figure 2.

Validation	Split				
	1	2	3	4	5
1	Testing	Training			
2	Training	Testing	Training		
3	Training		Testing	Training	
4	Training			Testing	Training
5	Training				Testing

Figure 2. Dataset Distribution for Validation

In the first validation, the first dataset is used as the testing data, while the second until fifth datasets are training data. In the second validation, the second dataset is used as testing data, and the other as training data. Validation is repeated until all datasets have been used as testing data.

Validation results are used to measure model performance. To measure the performance of the model used the confusion matrix. A confusion matrix is a useful tool for analyzing how well classifiers can recognize tuples/features of different classes[12]. Confusion matrix also provides performance appraisal of classification models based on the number of objects predicted correctly and incorrectly[13].

The dataset used for prediction of software defects is generally unbalanced because the amount of defect data is far less than that for non-defects [14]. Models that use unbalanced data cannot be evaluated through accuracy values [15]. Prediction models that are trained using unbalanced classes tend to produce majority class predictions [16] because they don't recognize the minority class well. For the evaluation of software defect prediction models generally use AUC (Area Under The Curve)[17].

3. Results and Discussion

The results of the application of the model and dataset obtained the value of performance-based on accuracy and AUC. The results of performance measurements are then visualized using the graph shown in Figure 3 and Figure 4. Figure 3 is a performance graph based on accuracy, while Figure 4 is a performance graph based on AUC. Both graphs show that the application of dimensional reduction algorithms can improve model performance both from accuracy and AUC.

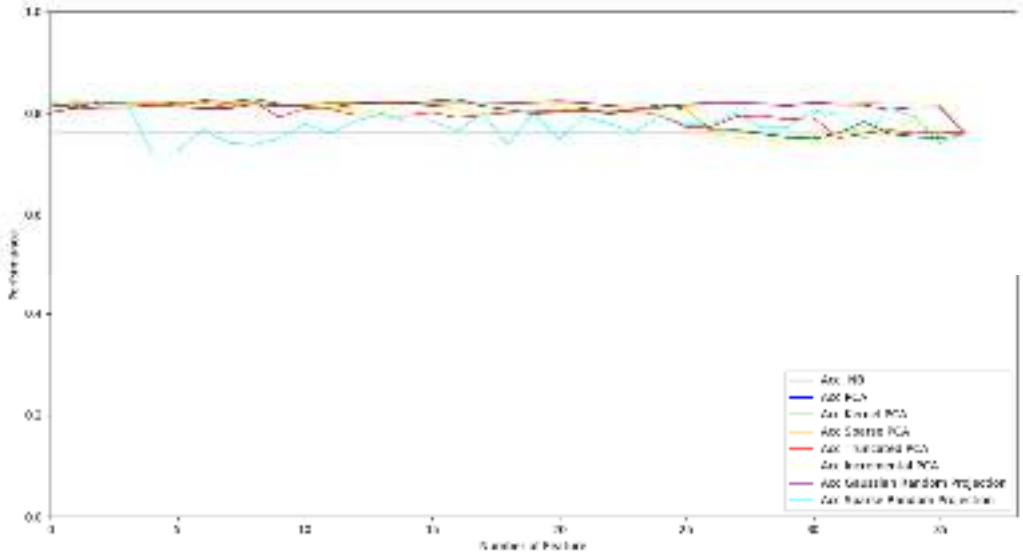

Figure 3. Accuracy of model

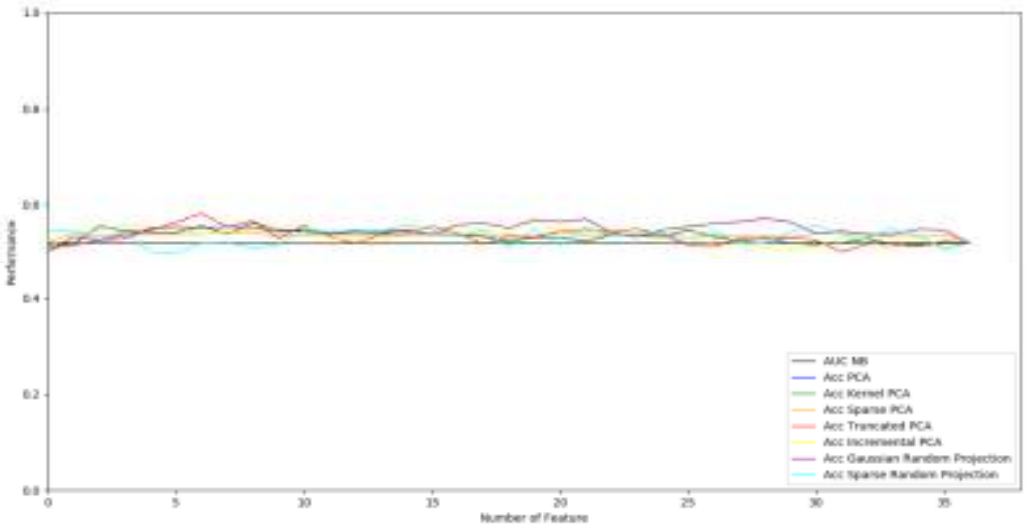

Figure 4. AUC (Area Under the Curve) of model

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 for Accuracy and Tabel 2 for AUC.

Table 1. P-value comparison of accuracy

Model	DT	PCA	Kernel PCA	Sparse PCA	Truncated SVD	Incremental PCA	Gaussian Random Projection	Sparse Random Projection
DT	1.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.001**
PCA	**0.000**	1.000	1.000	**0.000**	0.485	0.623	**0.002**	**0.001**
Kernel PCA	**0.000**	1.000	1.000	**0.000**	0.485	0.623	**0.002**	**0.001**
Sparse PCA	**0.000**	**0.000**	**0.000**	1.000	**0.000**	**0.000**	0.258	**0.000**
Truncated SVD	**0.000**	0.485	0.485	**0.000**	1.000	0.892	**0.000**	**0.002**
Incremental PCA	**0.000**	0.623	0.623	**0.000**	0.892	1.000	**0.000**	**0.004**
Gaussian Random Projection	**0.000**	**0.002**	**0.002**	0.258	**0.000**	**0.000**	1.000	**0.000**
Sparse Random Projection	**0.001**	**0.001**	**0.001**	**0.000**	**0.002**	**0.004**	**0.000**	1.000

Table 2. P-value comparison of AUC (Area Under the Curve)

Model	DT	PCA	Kernel PCA	Sparse PCA	Truncated SVD	Incremental PCA	Gaussian Random Projection	Sparse Random Projection
DT	1.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
PCA	**0.000**	1.000	1.000	0.258	0.981	0.612	**0.000**	0.851
Kernel PCA	**0.000**	1.000	1.000	0.258	0.981	0.612	**0.000**	0.851
Sparse PCA	**0.000**	0.258	0.258	1.000	0.432	0.827	**0.000**	0.271
Truncated SVD	**0.000**	0.981	0.981	0.432	1.000	0.678	**0.001**	0.856
Incremental PCA	**0.000**	0.612	0.612	0.827	0.678	1.000	**0.003**	0.540
Gaussian Random Projection	**0.000**	**0.000**	**0.000**	**0.000**	**0.001**	**0.003**	1.000	**0.000**
Sparse Random Projection	**0.000**	0.851	0.851	0.271	0.856	0.540	**0.000**	1.000

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 3 for Accuracy and Tabel 4 for AUC. All of models that implement dimensional reduction are significantly different.

Table 3. Significantly different comparison of accuracy

Model	DT	PCA	Kernel PCA	Sparse PCA	Truncated SVD	Incremental PCA	Gaussian Random Projection	Sparse Random Projection
DT	Not	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**
PCA	**Sig**	Not	Not	**Sig**	Not	Not	**Sig**	**Sig**
Kernel PCA	**Sig**	Not	Not	**Sig**	Not	Not	**Sig**	**Sig**
Sparse PCA	**Sig**	**Sig**	**Sig**	Not	**Sig**	**Sig**	Not	**Sig**
Truncated SVD	**Sig**	Not	Not	**Sig**	Not	Not	**Sig**	**Sig**
Incremental PCA	**Sig**	Not	Not	**Sig**	Not	Not	**Sig**	**Sig**
Gaussian Random Projection	**Sig**	**Sig**	**Sig**	Not	**Sig**	**Sig**	Not	**Sig**
Sparse Random Projection	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	Not

Table 4. Significantly different comparison of AUC (Area Under the Curve)

Model	DT	PCA	Kernel PCA	Sparse PCA	Truncated SVD	Incremental PCA	Gaussian Random Projection	Sparse Random Projection
DT	Not	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**
PCA	**Sig**	Not	Not	Not	Not	Not	**Sig**	Not
Kernel PCA	**Sig**	Not	Not	Not	Not	Not	**Sig**	Not
Sparse PCA	**Sig**	Not	Not	Not	Not	Not	**Sig**	Not
Truncated SVD	**Sig**	Not	Not	Not	Not	Not	**Sig**	Not
Incremental PCA	**Sig**	Not	Not	Not	Not	Not	**Sig**	Not
Gaussian Random Projection	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	**Sig**	Not	**Sig**
Sparse Random Projection	**Sig**	Not	Not	Not	Not	Not	**Sig**	Not

To find out the significant difference towards better or decreasing visualization using a boxplot diagram as shown in Figure 5 and Figure 6. Both Figure shows that the seven dimensional reduction models can significantly increase the accuracy and AUC of Decision Tree classifiers.

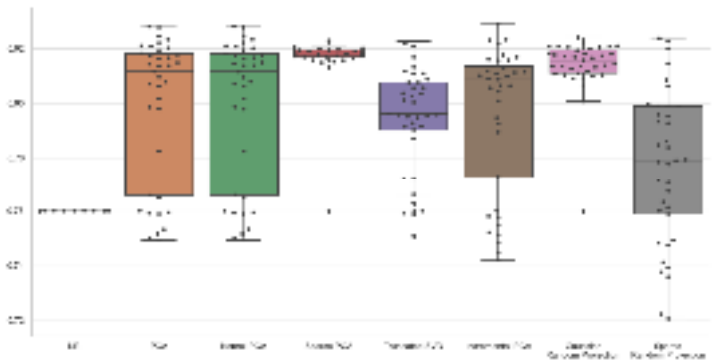

Figure 5. Boxplot of accuracy models

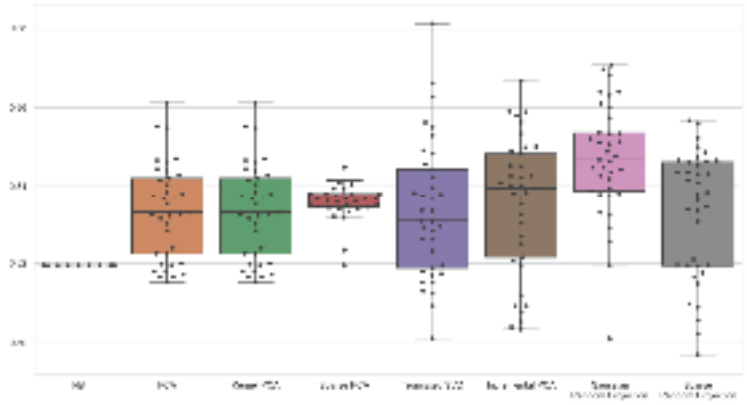

Figure 6. Boxplot of AUC (Area Under the Curve) models

4. Conclusion

Based on statistical analysis using ANOVA on the value of Accuracy and AUC, all the proposed models have significantly different from Naïve Bayes model. The visualization using boxplot diagram show that all proposed models have higher performance, so it can conclude that all proposed models have better performance. It's can state that the proposed models can find software defects better than Naïve Bayes.

References

[1] 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

[2] Malhotra R and Kamal S 2017 Tool to handle imbalancing problem in software defect prediction using oversampling methods *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* vol 2017-Janua (IEEE) pp 906–12

[3] Ni C, Chen X, Wu F, Shen Y and Gu Q 2019 An empirical study on pareto based multi-objective feature selection for software defect prediction *J. Syst. Softw.* **152** 215–38

[4] 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

[5] Aleem S, Capretz L F and Ahmed F 2015 Benchmarking Machine Learning Techniques for Software Defect Detection *Int. J. Softw. Eng. Appl.* **6** 11–23

[6] Pak C, Wang T T and Su X H 2018 An Empirical Study on Software Defect Prediction Using Over-Sampling by SMOTE *Int. J. Softw. Eng. Knowl. Eng.* **28** 811–30

[7] Rhmann W, Pandey B, Ansari G and Pandey D K 2019 Software fault prediction based on change metrics using hybrid algorithms : An empirical study *J. King Saud Univ. - Comput. Inf. Sci.* 4–9

[8] 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

[9] Limsettho N, Bennin K E, Keung J W, Hata H and Matsumoto K 2018 Cross project defect prediction using class distribution estimation and oversampling *Inf. Softw. Technol.* **100** 87–102

[10] Shepperd M, Song Q, Sun Z and Mair C 2013 Data quality: Some comments on the NASA software defect datasets *IEEE Trans. Softw. Eng.* **39** 1208–15

[11] Rahmadani S, Dongoran A, Zarlis M and Zakarias 2018 Comparison of Naive Bayes and Decision Tree on Feature Selection Using Genetic Algorithm for Classification Problem *J. Phys. Conf. Ser.* **978**

[12] Jiawei H, Kamber M, Han J, Kamber M and Pei J 2012 *Data Mining: Concepts and Techniques*
[13] Gorunescu F 2011 *Data mining: concepts and techniques*
[14] Ryu D and Baik J 2016 Effective multi-objective naïve Bayes learning for cross-project defect prediction *Appl. Soft Comput. J.* **49** 1062–77
[15] Catal C 2012 Performance evaluation metrics for software fault prediction studies *Acta Polytech. Hungarica* **9** 193–206
[16] Khoshgoftaar T M, Gao K and Seliya N 2010 Attribute selection and imbalanced data: Problems in software defect prediction *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI* **1** 137–44
[17] Tantithamthavorn C and Hassan A E The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models 1–20