Weighted knn using grey relational analysis for cross-project defect prediction

DI Ulumi and D Siahaan
Informatics Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
E-mail: daniel@if.its.ac.id

Abstract. Defect prediction plays important roles in detecting vulnerable component within a software. Some researchers have tried to improve the accuracy of software defect prediction so that it helps developer to manage resources (human, cost, and time) better. They focus on building the software defect prediction model only for a specific domain. To our knowledge, research on cross-project domains has not been carried out before. This research developed method to predict software defect for cross-project domains. Thus, the domain contains datasets with different number of features. To extend shortered features in a dataset, the method calculates the missing values. This research employed weighted KNN to fill in the missing value. The refilled datasets were then classified using naïve bayes and random forest. This research also conducted a feature selection process to select relevant features for detecting defects by means of a comparative analysis of methods of selection of features. For the experimentation, this research used seven NASA public dataset MDPs. The results show that for imbalance data, naïve bayes combined with information gain (IG) or symmetric uncertainty (SU) feature selection produced the best balance, i.e. 0.4975. It also shows that for balance data, random forest combined with gain ratio (GR) produced the best balance, i.e. 0.7795. In general, the developed method performed relatively alike the previous method, which classify only specific domain, i.e. 0.4975. It even outperformed previous method for dataset PC2, i.e. 0.4033.

1. Introduction
Software defect prediction model can predict defect prone modules that contain measurements for identifying the potential defect modules. Software defect prediction activities is part of software quality discipline [1]. The prediction method estimates the number of defects and locates the position of each defect [2]. It helps project manager, developer, and tester in identifying any defect prone modules within a software [3]. Conducting software defect prediction at the end of testing phase may not be beneficial. It would spend huge amount of money and effort in order to achieve target software quality [4][5]. There are several methods to solve the software defect prediction model, e.g. Greedy Forward Selection (GFS), ensemble learning classification technique, and Average Probability Ensemble (APE) [6]. APE is a classification technique that calculates the mean of defects features. This method uses six datasets NASA public MDP and resolves imbalance data and redundant features. But, this method was designed to handle a specific project domain. Another method is Defect Prediction uses Relational Association Rule (DPRAR) [7]. This work comprises of three steps. There are pre-processing, calculating spearman rank correlation coefficients, and training a model by using a DPRAR algorithm. It focuses on identifying the relationship between two software features. Relevant features are used at the next step, the model testing phase that has been built will be tested per dataset.
But, the model can only be used for specific datasets. Another method uses manual record for selection feature and naïve bayes for classification [8]. It can resolve defect and give good result using naïve bayes classification. But the software defect prediction model can only be used on a limited number and simple prediction metrics. Another study describes the rejoELM (ELM with reject option). It aims to avoid misclassification. The IrejoELM (Imbalance ELM with reject option) shows the best performance for imbalance data [9]. However, this method has deficiency in term of determining the weight that still uses random values. It can only be used on binary-based classifications. The model can only be used for specific project domain. Another method uses feature selection and data sampling to build software defect prediction models [10]. Feature selection is based on seven feature selection method. It uses Synthetic Minority Over-Sampling Technique (SMOTE), and Random Under-Sampling (RUS) for data sampling. Their experiment result shows that data balancing using SMOTE is more effective than RUS for modeling a software defect prediction. Nevertheless, the model was built only for specific project domain.

Another method uses five selection methods and Cluster Based Classification (CBC) with two NASA public MDP datasets [11]. Each data has a domain. Each domain has an amount of different features and similar features. Five feature selection methods are gain ratio (GR), information gain (IG), one-r (OR), relief-f (RFF) and symmetric uncertainty (SU). Feature selection can provide information feature combination which results in better performance. This method can improve the accuracy of software defect prediction that outperformed previous methods. The results obtained by a combination of CBC classification and IG feature selection produce a good result compared to other feature selection methods. So, the proposed method can improve the accuracy of software defect prediction that outperformed previous methods. However, this method is less efficient, because the processing of each dataset is done one-on-one with different software prediction models according to the number of features of each dataset. Therefore, it is necessary to model the software defect prediction for cross-project with the different number of features.

In this paper, we propose a new software defect prediction model on cross-project domains with different set of features. To solve the difference in the number of features, this study adopted weighted KNN for generating any missing value. After filling all missing value, the classification process would be carried out using a classification method. This study experimented two different classification methods, i.e. naïve bayes and random forest. The goal is to alleviate the cost, people and time in a project because the testing process is focused on the most vulnerable modules of the defect. The model would allowed software defect prediction on cross-project domains.

2. Previous Research

Research work by [6] using GFS feature selection method and the ensemble learning classification technique, APE learning for the software defect prediction model. The test result shows the highest accuracy value found from combination GFS and APE learning. But, the software defect prediction model is built in the specific dataset. The software defect prediction model has been proposed by [7]. However, the model can only be used for specific datasets. Another method uses manual record for selection feature and naïve bayes for classification [8]. But the software defect prediction model can only be used on a number of metrics that are few and simple predictors. Another method describes the rejoELM that can be helpful in avoiding misclassification and IrejoELM that can show the best classification in imbalance data [9]. But, the model can only be used for specific datasets. In addition, feature selection and data sampling are also performed on research work by [10], in this research show that SMOTE is more effective in a model software defect prediction. But, the software defect prediction model is built in the specific dataset. Another method uses five selection methods and CBC with two datasets NASA public MDP [11]. Five selection methods are GR, IG, OR, RFF and SU. The result shows that CBC with feature selection is higher than CBC without feature selection. However, it is done each dataset one-on-one. So, the software defect prediction model is less efficient.

3. The Proposed Method
Software defect prediction in this paper was described step by step. The steps can be seen in Figure 1. There are six steps that are conducted to obtain the software defect prediction model. The first step is sequence feature selection. In this stage, seven data NASA public MDP are ordered based on an amount of features. Before ordering, redundant data is removed from each dataset. Redundant data is data that has the same features and class values with previous data [11] [12].

The second step is weighted KNN. There are four steps in the weighted KNN that are grouping data, calculating the closest nearest neighbor, calculating Grey Relational Grade (GRG), and filling missing values as shown in figure 2. Before grouping the data, it was divided non-defect data and defect data. Then each non-defect and defect data are divided into complete data and incomplete data.

Then, one closest neighbor is calculated. The value of one closest neighbor is obtained by equation 1.

\[
distance (x_i, x_j) = \sqrt{\sum_{l=1}^{n} (x_{il} - x_{jl})^2}
\]

where \(x_i\) is incomplete instance and \(x_j\) complete instance. Each incomplete instance is inputted instance of complete data that has the minimum value of one closest neighbor.

Then, the data are normalized [13]. Each feature is scanned to find the minimum and maximum values. These values are used to calculate normalization using equation:

\[
x_i(j) = \frac{x_i(j) - \min_{l=1}^{n} [x_l(j)]}{\max_{l=1}^{n} [x_l(j)] - \min_{l=1}^{n} [x_l(j)]}
\]

where \(x_i(j)\) is data \(i\) feature \(j\), \(\min_{l=1}^{n} [x_l(j)]\) is minimum value of each feature and \(\max_{l=1}^{n} [x_l(j)]\) is maximum value of each feature. Where \(x'_i(j)\) is feature \(j\) which has a missing value. Then calculate the distance with a matrix. The equation is defined as follows:

\[
\Delta_{ai} (j) = x'_i(j) - x'_i(j)
\]
where $\Delta_{oi}(j)$ is distance incomplete instance and complete instance after normalization and $x'_i(j)$ is the value after normalization. Grey relational coefficient is calculated to know the relationship of the ideal and actual experimental results with the equation:

$$GRC\gamma_{oi}(j) = \frac{\Delta_{min} + \rho \Delta_{max}}{\Delta_{oi}(j) + \rho \Delta_{max}}$$  

(4)

$GRC\gamma_{oi}(j)$ is grey relational coefficient, $\rho (\rho \in [0,1])$ is a commonly defined coefficient $\rho = 0.5$ [14] [13], $\Delta_{min}$ is minimum value on $\Delta_{oi}(j)$ and $\Delta_{max}$ is maximum value on $\Delta_{oi}(j)$. Then grey relational grade is calculated with mean value of grey relational. The equations is defined as follows:

$$GRG(Y, X_i) = \frac{1}{m} \sum_{k=1}^{m} GRC\gamma_{oi}(j)$$  

(5)

where $m$ is amount of feature. The higher $GRG(Y, X_i)$ the correlation between $Y$ and $X_i$ is getting stronger. The stronger the correlation, the greater the weight. In most cases, the weight of each nearest neighbor is defined as follows:

$$w_j = \frac{1}{d_j}$$  

(6)

where $d_j$ is distance instance $j$ and target instance $i$.

Filling missing value is obtained by equation:

$$x_{ip} = \frac{\sum_{j=1}^{k} w_j x_{jp}}{\sum_{j} w_j},$$  

(7)

where $x_{ip}$ is missing value form $X_i$ instance

---

**Figure 2.** The steps involved in weighted KNN.

The third step is $k$ cross-fold validation. In this stage, ten cross-fold validation is used. Then, the fourth step is naïve bayes and random forest classification. Balance value is obtained from naïve bayes and random forest classification. If balance value is the same or down, the process is ended.

The fifth step is feature selection using IG, GR, OR, SU, and RFF. The feature selection process for each algorithm produces a ranking feature. Feature selection is done to show which features are relevant. The higher ranking features show the more relevant. A number of Top N features are taken from the highest feature rating values that are used to enter the next process. Determination of the value of N based on iteration [4]. The number of Top N features that can produce the highest predictive value in the classification process that will be used as a result. The last step is naïve bayes and random forest classification. The process is the same with fourth step.

### 4. Result and Discussion

The datasets used in this study are KC3, MW1, CM1, PC1, PC2, PC3 and PC4. These dataset were obtained from NASA public MDP. There are 6293 data that consist of 545 defect class and 5748 non-defect class. The evaluation uses balance value, where the higher the value, the better the quality of software defect model.
The experiments in this study were conducted by doing two steps testing scenario. In the first step, we focused on generating any missing value. There were three testing scenarios which were done to determine the missing values, namely:
1. Calculate the average of nearest neighbor, get the minimum value of the calculation using k = 2 up to k = 10
2. Calculate the average of nearest neighbor, and then find the minimum value of all the results using k = 2 up to k = 10 and with additional feature selection.
3. Get the minimum value of the result using the one closest neighbor and feature selection.

To get the k value of all three testing scenarios, we calculate all 6293 data using (1)-(7) in Section 3. Table 1 shows the result after the three testing scenarios were carried out. The comparison among the three scenarios was generated from Step 3 until Step 6 in Section 3.

| Scenario | 1st | 2nd | 3rd |
|----------|-----|-----|-----|
| Balance Value | 0.495 | 0.4763 | 0.4975 |

Based on Table 1, the best result is the third scenario. It means that generating the missing values using the closest neighbor and selecting features using feature selection produced the best balance value. The results show that imbalance data was solved using naïve bayes combined with IG or SU feature selection present the best balance value, shown in 3rd scenario in Table 1 (0.4975). The result of our research in accordance with [11] which shows the selection of the feature IG provides the best performance and can also increase the balance value. Not only the selection of the feature IG, but also Naïve bayes implemented in imbalance data will give the best performance compared to other classification techniques in [15].

Next, the second step consists of two testing scenarios which use imbalance data and balance data using the best result of the first step. In balance data, the combination of GR and random forest give the best result namely 0.7795 as shown in Table 2. This result in accordance with [10] which shows that balance data processed using random forest in NASA public MDP dataset present a better performance rather than imbalance data. So, based on our experiments and previous research done by Y H Wang [16], the equation of IG, SU and GR are similar to each others, so that they will also have similar value.

To provide in detail explanation about the performance of the proposed method, we carried out further analysis. Table 3 shows the analysis data by comparing the performance of the method on cross project domain and specific project domain. The same datasets from NASA Public MDP were used for this purpose.

| Cross-Project | Specific Data |
|---------------|---------------|
| KC3 | MW1 | CM1 | PC1 | PC2 | PC3 | PC4 |
| Balance Value | 0.4975 | 0.623 | 0.743 | 0.607 | 0.647 | 0.403 | 0.761 | 0.784 |
Based on Table 3, the value of cross-project is better than the value of PC2. This happens because the PC2 has many number of missing values, i.e. 1406 data. Based on this experiment, we can state that our method for inputting the missing values can improve classification performance. However, in this case misclassification in missing value data is lower than the misclassification in non-missing value data.

Based on research done by M Zhu [5], the method of inputting the missing value using k=5 to k=10 gives a low value in Normalized Root Mean Square Error (NRMSE). And based on our experiments, we also find that in the first testing scenario of first step we have k=9 to give the best balance value. In conclusion, using our proposed method, inputting the missing value using the closest neighbor in third scenario of first step will give the best performance out of all scenarios.

5. Conclusion

In this work, we proposed a method to produce software defect prediction model for cross-project domains. The method adopted weighted KNN for generating any missing value. The result shows that the proposed method can improve the performance of software defect prediction for cross-project domain. The use of the closest neighbor in the inputting process gives a better performance than the use of the average nearest neighbors (k = 2 to k = 10). The balance value of cross-project is better than PC2, which is 0.4975 compared to 0.4034. It is proven that not all features are used in this research that can time efficient. In the imbalance data, naïve bayes combined with IG or SU feature selection produced the best balance, i.e. 0.4975. The accuracy of the proposed method is relatively lower than previous method that uses specific data. Nevertheless, the method managed to create a classification model for cross-project domains that perform relatively similar to the model produced by previous method which is used only for specific domain. Moreover, the combination of GR and random forest contribute to the best classification performance, i.e. 0.7795 for the balance data.

References

[1] Ö. Faruk 2015 Software defect prediction using cost-sensitive neural network 33 263–77
[2] W. Li, Z. Huang and Q. Li 2016 Knowledge-based systems three-way decisions based software defect prediction Knowledge-Based Syst 91 263–74
[3] G. Abaei, A. Selamat and H. Fujita 2015 Knowledge-based systems an empirical study based on semi-supervised hybrid self-organizing map for software fault prediction Knowledge-Based Syst 74, 28–39
[4] H. B. Yadav and D. K. Yadav 2015 A fuzzy logic based approach for phase-wise software defects prediction using software metrics Inf. Softw. Technol
[5] R. Malhotra 2016 An empirical framework for defect prediction using machine learning techniques with Android software Appl. Soft Comput 49 1034–50
[6] I. H. Laradji, M. Alshayeb and L. Ghouti 2015 Software defect prediction using ensemble learning on selected features Inf. Softw. Technol 58 388–402
[7] G. Czibula, Z. Marian and I. G. Czibula 2014 Software defect prediction using relational association rule mining Inf. Sci. (Ny) 264 260–78
[8] P. He, B. Li, X. Liu, J. Chen and Y. Ma 2014 An empirical study on software defect prediction with a simplified metric set 59 170–90
[9] D. P. P. Mesquita, L. S. Rocha, J. Paulo, P. Gomes and A. R. Rocha 2016 Classification with reject option for software defect prediction 49 1085–93
[10] K. Bashir, T. Li and C. Wondaferaw 2017 Enhancing Software Defect Prediction Using Supervised-Learning Based Framework 1-6
[11] F. P. B. Muhamad, D. O. Siahaan and C. Fatichah 2018 Software fault prediction using filtering feature selection in cluster-based classification IPTEK J. Proc. Ser 4
[12] F. Pralienka, B. Muhamad, D. O. Siahaan and C. Fatichah 2017 Perbaikan prediksi kesalahan perangkat lunak menggunakan seleksi fitur dan cluster-based classification J. Nas. Tek. Elektro dan Teknol. Inf. 6 275–83

[13] K. Vatansever and Y. Akgül 2018 Performance evaluation of websites using entropy and grey relational analysis methods: The case of airline companies Decis. Sci. Lett 7 119–30

[14] M. Zhu and X. Cheng 2015 Iterative KNN imputation based on GRA for missing values in TPLMS Proc. 2015 4th Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 94-9

[15] N. Maisarah, M. Sobran, A. Ahmad and Z. Ibrahim 2013 Classification of imbalanced dataset using conventional naïve bayes classifier 25–6

[16] Y. H. Wang and I. C. Wu 2009 Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms Softw. - Pract. Exp 39 701-36