Weighted k Nearest Neighbour Using Grey Relational Analysis to Solve Missing Value

Desepta Isna Ulumi and Daniel Siahaan

Abstract— **Defect in software takes the form of error, bug, fault, or failure. Predicting defect in software helps to improve the software quality. It helps developer identifying vulnerability within the software component earlier. Researchers tried to enhance the performance of the defect prediction method to manage the project resources better. Previous researches applied the method on distinctive project domain. The problem is that a model can only be applied not after it provides sufficient software defect historical data of the given project domain. A model is only relevant for a specific project domain. This paper introduces an approach to build a generic model using a merged dataset of various project domains. Each dataset has originally different features. All missing value which is produced due to the merging of datasets of varying feature numbers should be calculated. We applied Weighted k-Nearest Neighbor (WkNN) and Grey Relational Analysis to calculate the missing values of a dataset. After all missing values have been filled in, we applied Naïve Bayes in order to classify the selected features. In the experimentation, we exercised on four different feature selection methods to find the most relevant features for all datasets. The results on seven empirical datasets indicate that by applying Naïve Bayes on selected presented selected by either Information Gain (IG) or Symmetric Uncertainty (SU), the best balance value can be obtained.**

Keywords— Gray relational analysis, naive bayes, software defect, weighted kNN.

I. Introduction

Software defect that may arise during software development can be foreseen using the classification method. Researchers build a prediction model using features extracted from its source code [1]. The features cover a wide range of software metrics, such as LOC count, Halstead attributes, McCabe attributes, control flow attributes, and commentary attributes.

There are a number of problems exist. First, not all metric software features are relevant in classifying whether a respected module is defect-prone or defect-free. Second, not all project domains logged all software metrics. Third, there is a significantly fewer number of defect-prone modules compare to defect-free modules. Finally, existing prediction models project domain sensitive. It means that a software defect prediction model is suitable only for a specific project domain.

There have been a number of efforts carried out to provide a prior solution. Laradji et al. used Greedy Forward Selection (GFS) feature selection method and the Ensemble Learning Classification technique [2]. This solution uses six datasets from NASA public MDP. It resolves the problem related to imbalance data and redundant features. Ensemble Learning Classification is a method for classifying data. It calculates the mean from some other classification techniques. It works well on an imbalance dataset. Nonetheless, the prediction model built from this solution was designed for a specific project. It would be insensitive when it tries to classify dataset of a different project.

Czibula et al. proposed a solution which uses Relational Association Rule (DPRAR) for classifying defect-prone module [3]. This solution uses three main processes. First, the solution pre-processes the dataset. The goal is to establish the dependencies between the features and the target output. Second, the solution calculates the spearman rank correlation coefficients. This process basically deletes any feature which is irrelevant in classifying defect-prone module. Finally, the solution trains the software defect prediction model. It uses a DPRAR algorithm. It focuses on distinguishing the relationship between two relevant features. For each relevant feature included in the next process, the testing process of the prediction model should be exercised per project domain. Czibula et al. analyzed the results on several accuracy measurements. They are the probability of detection (pd), specificity, precision, and area under the ROC curve (AUC) area. Just like the aforementioned solution, this solution also project domain sensitive. It only built specifically for a specific project domain.

Muhamad et al. improve the previous solution in term of accuracy. It used five popular feature selection method [4]. It uses Cluster-Based Classification (CBC) for classifying the defect-prone module. The study evaluates the solution on seven datasets from NASA public MDP. Each datum has a domain. Each domain has a number of different features and similar features. A feature is a property or characteristic in data which have various value, either from one object to another or from one time to another. Gain Ratio (GR), Information Gain (IG), One-R (OR), Relief-F (RFF) and Symmetric Uncertainty (SU) are categorized as five feature selection methods. These selection features generate the best performance in an information feature combination. This combination of method gives the best result rather than previous methods in terms of accuracy of software defect prediction. Combination of CBC classification and IG feature selection perform better compared with other combination methods.

Nevertheless, even though the combination of CBC classification and IG feature selection methods gives the best result, the combination method is less efficient in term of computation complexity. This is because the combination method carries out one-on-one processing. The process is based on a number of selected features of each dataset with various prediction models. Thus, it is important to build a prediction model that allows defect-prone module classification on generic datasets. The mode should be insensitive to a specific project domain.

Desepta Isna Ulumi and Daniel Siahaan are with Departement of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia. E-mail: daniel@if.its.ac.id

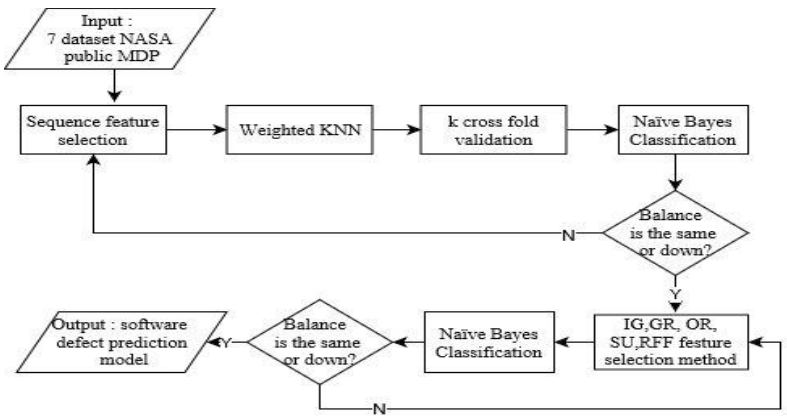
Input : 7 dataset NASA public MDP
Sequence feature selection
Weighted KNN
k cross fold validation
Naïve Bayes Classification
Balance is the same or down?
N
Y
IG,GR, OR, SU,RFF festure selection method
Naïve Bayes Classification
Balance is the same or down?
Y
N
Output : software defect prediction model

Figure. 1. Research method.

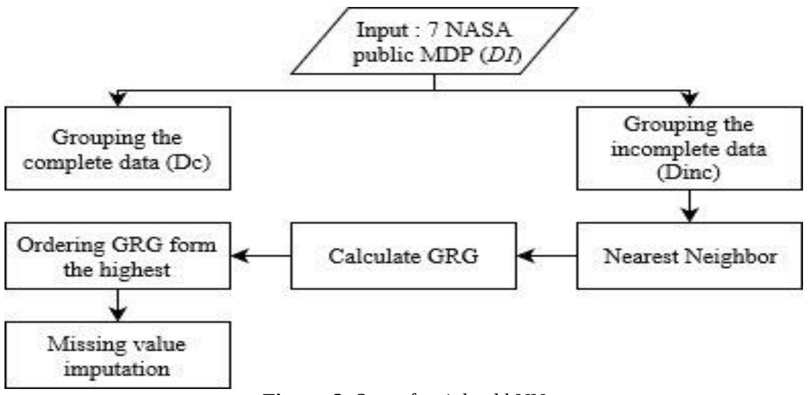
Input : 7 NASA public MDP (DI)
Grouping the complete data (Dc)
Grouping the incomplete data (Dinc)
Nearest Neighbor
Calculate GRG
Ordering GRG form the highest
Missing value imputation

Figure. 2. Step of weighted kNN

This study introduces a new software defect prediction model which is insensitive to a project domain. The built model was built on generic datasets, which have a various number of features. To solve the feature differences among merge datasets, this study proposes the use of Weighted k-Nearest Neighbor (WkNN). The method was used to fill the missing value produced as the result of merging the dataset of different project domain.

II. Method

In this paper, we describe the building of software defect prediction model in five separated processes. Nevertheless, the paper provides the overall view of our proposed solution (Figure 1.)

A. Sequence feature selection

In the first process, the researcher orders the feature that has less missing value after reducing redundant data. The redundant feature is a feature that has the same value and class [5]. The highest redundant feature is PC2.

B. Weighted k Nearest Neighbor (WkNN)ea

Nearest Neighbor (NN) is used to identify data points that are not yet classified [6]. Distance is evaluated from all training to testing data. The lowest distance value is called the nearest neighbor. The k-Nearest neighbor has some advantages such as easy to learn, resistance to noisy training data, effective if the training data is large [6]. But the k-Nearest Neighbor (kNN) method has memory limitations, complex computing, slowly running process and gullible with irrelevant features. This technique is easy to implement, but the k value affects the result. So T. Bailey and A. K. Jain modified the kNN by weighting and named weighted kNN (wkNN). WkNN is a method that evaluates the distance based on the value of k and the weight of each calculated value. The advantage of wkNN is to overcome the limitations of kNN by adding weight to each k, using all training samples not just k values, and suitable to be implemented in all datasets. In general, the weighted kNN process can be seen in figure 2. Data are divided into two groups, complete and incomplete data. The next step is calculating the nearest neighbor that calculate the distance of complete and incomplete data. Nearest neighbor is obtained by equation (1):

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

where x_i is an incomplete instance, and x_j is a complete instance. Each incomplete instance is obtained by equation (2):

$$x_{ip} = \frac{\sum_{j=1}^{k} x_{jp}}{k} \tag{2}$$

where x_{ip} target instance X_i, p is feature p in instance X_i and top k are based on $\{X_1, X_2, \ldots, X_k\}$. According to [7], before entering the next process, the data needs to be normalized. Normalized is obtained by equation (3):

$$x_i(j) = \frac{x_i(j) - min_{i=1}^{n}[x_i(j)]}{max_{i=1}^{n}[x_i(j)] - min_{i=1}^{n}[x_i(j)]}. \tag{3}$$

TABLE 1.
WEIGHTED KNN USING DIFFERENT K

A	B	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
19		0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429
20		0.446	0.4437	0.4437	0.4459	0.4459	0.4459	0.4459	0.4459	0.4459
21	6	0.4425	0.4424	0.4424	0.4443	0.4446	0.4446	0.4446	0.4526	0.4445
22	7								0.4658	0.4428
23	15								0.4659	
24	22								**0.4806**	
25	23								0.4664	

Notes: A = amount of features, B = number of feature

TABLE II.
BALANCE VALUE OF IG METHOD

	R1	R2	R3	R4	R5	R6
	feature 12	R1 + feature 22	R2 + feature 1	R3 + feature 21	R4 + feature 24	R5 + feature 18
Balance value	0.3613	0.4144	0.4544	0.4876	**0.4906**	0.4905

TABLE III.
BALANCE VALUE OF SU METHOD

	R1	R2	R3	R4	R5	R6
	feature 22	R1 + feature 12	R2 + feature 24	R3 + feature 21	R4 + feature 1	R5 + feature 18
Balance value	0.3585	0.4144	0.4282	0.4637	**0.4906**	0.4905

Where $x_i(j)$ is data *i* feature *j*, $min_{i=1}^{n}[x_i(j)]$ is the minimum value of each feature, and $max_{i=1}^{n}[x_i(j)]$ is the maximum value of each feature. Where $x'_0(j)$ is feature *j* and shows the greatest value of each feature. Then calculate the distance with matrix form as we can see an equation (4):

$$\triangle_{oi}\,(j) = x'_o(j) - x'_i(j) \quad (4)$$

where $\triangle_{oi}\,(j)$ is incomplete distance 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 as we can see an equation (5):

$$GRC\ \gamma_{oi}(j) = \frac{\triangle_{min}+\rho\triangle_{max}}{\triangle_{oi}(j)+\rho\triangle_{max}}. \quad (5)$$

$GRC\ \gamma_{oi}(j)$ is grey relational coefficient, ρ (ρ €[0,1]) is a commonly defined coefficient ρ = 0.5 [7][8], $\triangle_{min}$ is minimum value on $\triangle_{oi}\,(j)$ and $\triangle_{max}$ is maximum value on $\triangle_{oi}\,(j)$. Then grey relational grade is calculated with the mean value of grey relational as we can see an equation (6):

$$\boldsymbol{GRG(Y, X_i) = \frac{1}{m}\sum_{k=1}^{m} GRC\ \gamma_{oi}(j)} \quad (6)$$

where *m* is the 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 gain. In most cases, the weight of each nearest neighbor is defined as follows (7):

$$\boldsymbol{w_j = \frac{1}{d_j}}, \quad (7)$$

where $d_j$is distance instance j and target instance *i*. Filling missing value is obtained by equation (8):

$$\boldsymbol{x_{ip}} = \frac{\sum_{j=1}^{k} w_j x_{jp}}{\sum_j^k w_j}, \quad (8)$$

where x_{ip} is missing value form X_i instance

C. K Cross Fold Validation

In this study, ten cross-fold validation is used.

D. Naïve Bayes Classification

After filling the missing value, the next step is naïve bayes classification. The iteration can stop if the balance value n is less than the balance value n-1.

E. IG, GR, OR, SU, RFF Feature Selection Method

Some features from the previous step are selected that use five feature selection method. The approach of feature selection method used in this research filters. The filter is a feature selection method based on feature rank [5].

1) Information Gain

Information gain is one of the feature selection techniques that are able to assess the importance of features by measuring class-related [9]. Generally, the information gain can change the value of the uncertainty of information (entropy) into a measure of the value of information to be obtained. The value of information gain is obtained by equation (9):

$$IG(Class|Attribute = H(Class) - H(Class|Attribute \quad (9)$$

where *H* is the entropy. It is assumed that *A* is all features and classes dependent on all training. Example value (a, y) with $y \in Class$ defines the value of the specific instance for the feature $a \in A$, V represents the set of features i.e., $V = \{value(a,y)|a \in A \cap y \in Class\}$. The IG formula on each $a \in A$ feature is defined as follows (10):

$$IG(Class, a) = H(class) - \sum_{v\in V}\frac{\{y\in Class \vee value(a,y)=v\}}{|Class|} xH(y\{Class|value(a,y) = v\}) \quad (10)$$

2) Gain Ratio

Gain ratio modifies the information gain technique by taking into account the number of results obtained by the feature test condition [10]. The value of the gain ratio is obtained by equation (11):

$$GR(Class, a) = \frac{IG(Class,a)}{H(a)} \quad (11)$$

where $H(a)$ is obtained by equation (12):

$$H(a) = -\sum_{v \in V} \frac{|\{y \in Class \vee value(a,y) = v\}|}{|Class|}$$
$$x\ log_2 \frac{|\{y \in Class \vee value(a,y) = v\}|}{|Class|} \quad (12)$$

All the notations in the gain ratio formula are the same with IG.

3) One R

One R is built a feature called one rule for each feature in the dataset [5]. Algorithm One-R is defined as follows [11]:

```
For each feature f,
    For each value v from the domain of f
        Select an instance set with feature f having a value of v
        It is assumed that c is the class that has the highest frequency
        Apply "if feature f has value v then the class is c" for feature f
Output rules with the highest classification accuracy.
```

4) Symmetric Uncertainty

Symmetric Uncertainty (SU) also compensates the IG bias against features with a more different value and normalizes the value in range 0 to 1 [10]. The value of symmetric uncertainty is obtained by equation (13):

$$SU(Class, a) = 2x \frac{IG(Class,a)}{H(Class)+H(a)} \quad (13)$$

The equation is similar to IG and GR.

5) Relief f

Relief F is a feature selection techniques which evaluate several times and gives weighted value for each feature based on feature ability to differentiate each class and get the features which the weighted value fulfill the threshold value according to relevan features [11]. The Relief F algorithm is shown below:

```
Input:   a training set D, the number of iteration m, the number of nearest neighbors k,
         the number of features n, predefined feature weight threshold δ.
Output:  feature subset S constituted by features whose weights are all greater than
         the weight threshold δ.
Step 1:  Let S=∅, set all feature weights W(Ft)=0, t = 1,2,...,n.
Step 2:  For j=1 to m do
             (1) select a sample R from D randomly.
             (2) find out k nearest neighbors Hi (i = 1,2,...,k) from the same class
                 and k nearest neighbors Mi(C) (i = 1,2,...,k) from each different class C.
             (3) (3) For t=1 to n do
                 $W(F_t) = W(F_t) - \sum_{i=1}^{k} \frac{diff(F_t,R,H_i)}{mk} + \sum_{C \notin ClassR}$
Step 3:  For t=1 to n do
             If $W(F_t) > \delta$ then add feature $(F_t)$ to S
             In (1), P(C) is the probability distribution of class C , Class(R) is the
             category R belongs to, Mi(C) denotes the i Near Miss of R in class C,
             diff(Ft,R1,R2) denotes the difference between R1 and R2 on Ft. If Ft is discrete:
                 $diff(F_t, R_1, R_2) = \begin{cases} 0; R_1[F_t] = R_2[F_t] \\ 1; R_1[F_t] \neq R_2[F_t] \end{cases}$
             If Ft continues:
                 $iff(F_t, R_1, R_2) = \frac{|R_1[F_t] - R_2[F_t]|}{max[F_t] - min[F_t]}$
```

The last step is naïve-bayes classification, and the process is the same as the fourth step.III. Results and Discussion

This research is applied in seven NASA public MDP datasets, and the total data used is 6293. Then to evaluate, ten cross fold validation is implemented. The training, which is the process for building the software defect prediction model, can be seen in Figure 1. Meanwhile, testing is the process from the result of training which is classified using Naïve Bayes. Each test result on each fold is measured by confusion metrics that can be seen in Table IV.

TABLE IV.
CONFUSION MATRIX

Prediction	Actual	
	Defect	Non-defect
Defect	TP	FP
Non-defect	FN	TN

Probability of detection (pd) means all the successful values of the prediction systems in predicting the software defect, while the probability of false alarm (pf) means misclassification values of prediction systems in determining defect-free module as defect module [4]. The definition of Balance Value is the value which is available in the range of pd and pf. Balance value is called to give the best result if it is closer to 1. For measuring pd, pf and balance can be shown in equation (14), (15) and (16):

$$pd = \frac{\boldsymbol{TP}}{(\boldsymbol{TP+FN})} \quad (14)$$

$$pf = \frac{\boldsymbol{FP}}{(\boldsymbol{FP+TN})} \quad (15)$$

$$balance = \boldsymbol{1} - \frac{\sqrt{(\boldsymbol{0-pf})^2 + (\boldsymbol{1-pd})^2}}{\sqrt{\boldsymbol{2}}} \quad (16)$$

Balance value after filling the missing value can be seen in Table I. Based on Table I, the best of balance value is k equal 9, where the balance value is 0.4806. The result of our research supports research from Zhu and Cheng [8]. This is because the best result of k is between 5 and 10 in Normalized Root Mean Square Error (NRMSE). After filling the missing value, twenty-four features in k equal 9 were selected which will be used in five features selection methods.

Based on five features selection methods, the combination of Naïve Bayes and IG or SU give the best result, which is 0.4906 as shown in Table II and Table III. The result of our research, in accordance with [10], stated that the equation of IG, GR, SU are similar so that they have similar value too. Based on Table II, R1 is the first rank which has balance value 0.3585. We do the iteration continuously. If the value of Rn is higher than the value of Rn-1, the iteration is done until we get the value of Rn is less than the value of Rn-1. Based on Table II and Table III, the best value is in the fifth rank (R5) so to be processed to testing processes, we have five features; R1, R2, R3, R4, and R5. Both data in Table II and Table III are processed in the testing processes produced the testing value is 0.4959. So, we can conclude that using IG or SU have a similar testing value at the end of our experiments.

III. Conclusion

In this research, based on our experiments and analysis, Naïve Bayes with Information Gain (IG) and Symmetric Uncertainty (SU) feature selection presented the best balance value, which is 0.4959. It is proven that not all features are used in this research. In addition, our proposed method can also improve the performance of software defect prediction with the best result.

References

[1] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," vol. 59, pp. 170–190, 2015.

[2] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Softw. Technol.*, vol. 58, pp. 388–402, 2015.

[3] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Inf. Sci. (Ny).*, vol. 264, pp. 260–278, 2014.

[4] F. P. B. Muhamad, D. O. Siahaan, and C. Fatichah, "Software Fault Prediction Using Filtering Feature Selection in Cluster-Based Classification," *IPTEK J. Proc. Ser.*, vol. 4, no. 1, p. 59, 2018.

[5] F. Pralienka, B. Muhamad, D. O. Siahaan, and C. Fatichah, "Perbaikan Prediksi Kesalahan Perangkat Lunak Menggunakan Seleksi Fitur dan Cluster-Based Classification," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 6, no. 3, pp. 275–283, 2017.

[6] N. Bhatia and C. Author, "Survey of Nearest Neighbor Techniques," *IJCSIS) Int. J. Comput. Sci. Inf. Secur.*, vol. 8, no. 2, pp. 302–305, 2010.

[7] K. Vatansever and Y. Akgűl, "Performance evaluation of websites using entropy and grey relational analysis methods: The case of airline companies," *Decis. Sci. Lett.*, vol. 7, pp. 119–130, 2018.

[8] M. Zhu and X. Cheng, "Iterative KNN imputation based on GRA for missing values in TPLMS," *Proc. 2015 4th Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2015*, no. Iccsnt, pp. 94–99, 2016.

[9] S. A. Putri and Frieyadie, "Combining integreted sampling technique with feature selection for software defect prediction," *2017 5th Int. Conf. Cyber IT Serv. Manag. CITSM 2017*, pp. 1–6, 2017.

[10] Y. H. Wang and I. C. Wu, "Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms," *Softw. - Pract. Exp.*, vol. 39, no. 7, pp. 701–736, 2009.

[11] J. Novakovic, "The Impact of Feature Selection on the Accuracy of Naive Bayes Classifier," *18th Telecommun. Forum TELFOR*, vol. 2, pp. 1113–1116, 2010.