Exploring the Influence of Feature Selection Techniques on Bug Report Prioritization

Yabin Wang, Tieke He, Weiqiang Zhang, Chunrong Fang, Bin Luo*
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
*luobin@nju.edu.cn

Abstract

To improve software quality, developers often open a bug repository and allow users to find bugs, describe bugs in the form of bug reports and submit bug reports to the repository. Based on the description, testers assign a priority to each bug report. In the beginning the process of priority assignment is performed manually. With the increasing amount of bug reports, researchers introduced classification methods to assign priorities automatically with all the features considered. In this paper feature selection methods are introduced to improve the effect of bug report prioritization using classification models. The experimental results show that feature selection based on Information Gain and Pearson Correlation can improve the precision and recall for bug report prioritization on two models, i.e., SVM and Naive Bayes.

Keywords: Feature selection, Bug report prioritization, SVM, Naive Bayes

1 Introduction

Bug repositories of open software projects are often accessible to the public. Both users and developers can submit problems and suggestions on how to improve the software to the repository in the form of bug reports. A bug report is helpful for debugging in many ways. It creates a communication between bug finders and bug fixers and enables developers discuss how to fix a bug. The openness of the bug repositories makes more people being able to help to find more bugs for the developers, such that the quality of the software can be greatly improved. In order to use bug repository more effectively, bug reports should be managed. Reported bugs should be analyzed to determine whether they are valid or not, correct or not, and unique or not. This is called a bug triage process [1].

Manually validating large number of bug reports can be time-consuming and tedious. Not all the bug reports are regarded as important on the side of developers. Each bug report should be assigned a priority to indicate its importance, and this process is called bug report prioritization. Some bug reports are about meaningful bugs, while other bug reports maybe just suggestions of adding software functions. Bug reports that report meaningful bugs should have a higher priority. If the developers analyze the bug reports one by one, some important bug reports may be postponed.

The amount of bug reports are often very large. In the Wordpress project we studied, there are more than 20 thousand reports. The large amount of bug reports makes it very time consuming to manually assign priorities to bug reports. One way to solve the problem is to ask bug reporters assign the priority. But lacking the full knowledge of the whole project, bug reporters could not correctly assign priorities.

To correctly assign priority to the reports, a bug triager needs to analyze the bug description in the bug report to get the information about which component the bug may belong to, what type of the bug it is, its severity and whether it shall be solved immediately or can be postponed. According to these information, priorities are assigned to bug reports.

Currently, there have been many studies focusing on automatically prioritize bug reports. For example, DRONE [2] analyzed the textual description, author and product of bug reports to assign priorities. Kremenet et al. [3] checked the success and failure of a bug report to prioritize reports. Lamkanfi [4] used various classification algorithms such as Support Vector Machine (SVM) and Naive Bayes to assign priorities for bug reports. In their study textual descriptions were transformed to feature vectors, which were the inputs of classification models. However, the study of [1] tells that not all the features are important for bug report prioritization. Their results motivated us to adopt feature selection methods on bug report prioritization.

In this paper, the textual description of a bug report are first transformed into a feature vector, and then we use feature selection to build a subset of features that can represent the whole set. This subset can give us enough information for classification to prioritize bug reports. The subset could obtain better classification results. In this paper, 7 most popular feature selection approaches including *CfsSubset* (CFS), *Correlation* (CO), *GainRatio* (GR), *InfoGain* (IG), *OneR* (OR), *ReliefF* (RF) and *SymmetricalUncert* (SU) [5, 6] are considered. These 7 techniques cover two main feature selection techniques the wrapper methods and filter methods. The wrapper methods search the feature space and evaluate feature set to find the optimal feature subset. The filter methods evaluate single feature, score each feature and rank features according to the scores. We conducted ex-

10.18293/SEKE2017-073

periments to compare the feature selection techniques and studied the effects of the number of features selected, and we found that GR, IG and CO were more suitable for bug report prioritization.

In our experiment, we evaluated the effects of some popular feature selection techniques on bug report prioritization. Two large open source projects Trac [7] and Wordpress [8] were used.

The main contributions of this paper are summarized as follows:

- We are the first to introduce feature selection methods to find the features that are most relevant to priorities. Previous researches used all the features to classify bug reports but failed to study the important features that are related with priorities. Feature selection can find out the important features.
- We compare different feature selection techniques in bug report prioritization. The results show that Information Gain and Pearson Correlation based approaches achieve the best performance.
- We study the influence of the number of features on the feature selection techniques and find that one third to half of the features are optimal for feature selection.
- We conducted experiments on two large open projects with large amount of bug reports to validate our proposals.

The rest of the paper is organized as follows. Section 2 presents the related work. The introduction of classification is described in Section 3. The framework along with the classification model used is presented in section 4. The experiment setup, evaluation and experiment results are presented in section 5, 6 and 7. Finally we summarize the threats to validity and conclude our work in section 8 and 9.

2 Related Work

Menzies and Marcus were the first researchers that studied the bug report prioritization [9]. They analyzed the severity labels text and information of bug reports of NASA and generated 5 severity labels. They first extracted word tokens from text information of bug reports and then preprocessed the text to remove stop words and performed stemming. Then the words were transformed into feature vectors and then fed into a classification model.

Lamkanfi et al. [10] extended the work of Menzies and Marcus and studied various classification algorithms. They found that SVM and Naive Bayse were two most effective classification models. This motivated us to use these two models in our research. Khomh et al. [11] studied crash reports prioritization based on the frequency of the crashes. In this paper all types of bug reports were studied instead of only focusing on crash reports. All of existing approaches failed to emphasize the features that are relevant to priority. In this paper we introduce feature selection techniques on bug report prioritization to find the most relevant features.

3 Classification of Bug Reports

Classification of bug reports consists of two main processes. The first is training bug reports with priorities assigned correctly by triagers to build a learning model. Suppose that $R(r_1, r_2, ..., r_n)$ is a feature vector transformed from a bug report for training, $r_1, r_2, ..., r_n$ are n values of n features. Each feature is a word in the bug description, representing one dimension of information of the bug report. Each r_i represents the frequency of each word occurring in R. For each feature vector R_i, there is a special class label which represents the priority y_i of that bug report. The output of this step can be represented as a learning function.

$$y = f(R) \tag{1}$$

In function 1, $f(R)$ can be some regulations or some formulas. In the second process of testing $f(R)$ receives new bug reports with no priority labels. The output y is the predicted priority for the input bug report. In order to evaluate a classification algorithm, the bug reports with correct labels of priority will work as inputs such that precision and recall of the classification algorithm can be calculated.

4 Approach

Figure 1 shows the framework of our approach.

1 Preprocess bug reports: A commonly used text transformation method [12] is adopted to preprocess bug reports, and transform reports into feature vectors with each bug report transformed into one vector. In this phase, special symbols, brackets and punctuation are removed. Non-alphabetic words, common words and stop words are also removed, because these words are meaningless and unimportant. Stemming is then applied on the remained words to convert them into their ground meaning. The ground form of each word is seen as a feature. The frequency of occurring of each word in each bug report is generated as the value of a specific feature in a vector.

2 Split data: The feature vector set is divided into two sets, training set and testing set using five-fold cross validation [13].

3 Select features: Different feature selection techniques are applied on the training set to construct a subset of features. The outputs of feature selection are the new mapped feature vector set and the subset of features. The new mapped feature vector set only contains the information of constructed subset of features.

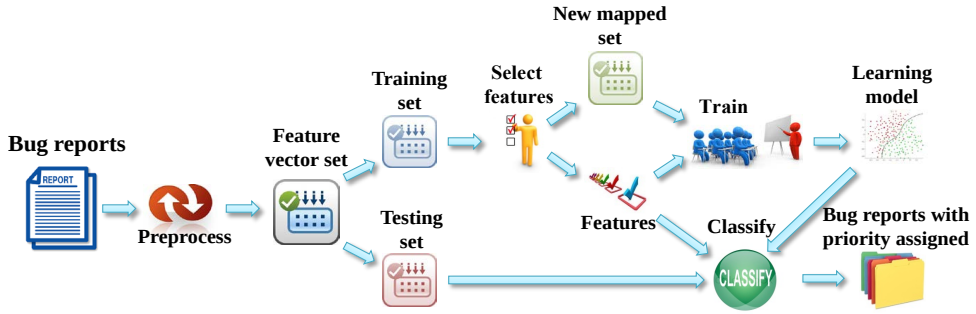

Figure 1: Framework

4 Train: Two classification algorithms, i.e., SVM and Naive Bayes are used for training the feature vectors. The output of training is the specific learning model.

5 Classify: The selected features and the learning model are used for classifying the testing set. The output is that each report is assigned a priority class.

4.1 Feature Selection for Bug Report Prioritization

There are mainly two sorts of feature selection methods, "wrapper" and "filter" methods [14]. Wrapper methods search for feature subsets of bug reports and find the subset with the highest quality [15]. Filter methods score for each feature of bug reports based on some criteria, independently. Then the features are ranked according to the score and top N features are selected [14]. In this paper 7 most popular feature selection methods *CfsSubset* (CFS), *Correlation* (CO), *GainRatio* (GR), *InfoGain* (IG), *OneR* (OR), *ReliefF* (RF) and *SymmetricalUncert* (SU) are studied. The description of the methods are as follows.

- *CfsSubset* (CFS): CFS uses Best First and Greedy Stepwise to search feature space and uses minimum description length (MDL) to measure the correlation inside a subset and correlation between the subset and the priority. The subset with the highest quality is selected [16].
- *Correlation* (CO): CO measures the Pearson Correlation between a feature and the priority to score for each feature [17].
- *InfoGain* (IG): IG scores a feature by measuring the Information Gain between a feature and the priority [18]. The concepts of Information Gain comes from information theory [19].
- *GainRatio* (GR): GR scores a feature by measuring the gain ratio with respect to the priority. GR is a measure extended from IG. The difference is that based on IG GR will further calculate the information generated by splitting the training data according to the different kinds of value of a feature [6].
- *OneR* (OR): OR evaluates a feature by measuring the classification accuracy by using OneR classifier [5].
- *ReliefF* (RF): RF evaluates a feature by repeatedly sampling a feature vector and considering the value of the given feature for the nearest vector of the same and different priority class [20].
- *SymmetricalUncert* (SU): SU evaluates a feature by measuring the symmetrical uncertainty with respect to the priority [21].

4.2 Feature Selection Based Naive Bayes

The Naive Bayes classifies bug reports by calculating the probability of a bug report assigned to a priority label using Bayes rule of conditional probability. The probability of one new bug report belonging to each priority is calculated and the priority with the highest probability is assigned to the new report.

The Naive Bays can be stated as equation 2. $P(C_i|R)$ indicates that given a report R, the probability that the priority C_i is assigned to R.

$$P(C_i|R) = (P(C_i) \times P(R|C_i))/P(R) \tag{2}$$

Naive Bayes maximizes $P(R|C_i) \times P(C_i)$ to find the priority with the highest probability for a bug report. The assumption of Naive Bayes is that a value of a feature does not depend on the value of other features. Therefore, $P(R|C_i)$ is calculated by equation 3.

$$P(R|C_i) = \prod_{j \in SF} P(r_j|C_i) \tag{3}$$

$$P(C_i|R) > P(C_j|R) \;\; j \neq i \tag{4}$$

If equation 4 is satisfied for each C_j, C_i is assigned to the new bug report. In equation 3 SF is the selected feature set. Feature selection techniques will adjust the value of $P(R|C_i)$ to affect the results of Naive Bayes by choosing an optimized SF.

4.3 Feature Selection Based SVM

SVM transforms the input feature vectors of bug reports to a higher dimension and then searches for a hyperplane with the maximum margin in the new mapped space [22]. A simple hyperplane can be defined as equation 5.

$$w \cdot x + b = 0, x = \begin{matrix} r_{11} & r_{1j} & ... & r_{1n} \\ r_{21} & r_{2j} & ... & r_{2n} \\ ... & ... & ... & \\ r_{m1} & r_{mj} & ... & r_{mn} \end{matrix}, j \in SF \quad (5)$$

In this equation, x are the feature vectors of bug reports that lie on the hyperplane, b is a scalar often referred as a bias, w is a weighting vector, $\cdot$ is the dot product, SF is the index set of selected features. Feature selection techniques adjust SF to change the hyperplane and will affect the results of SVM.

5 Experimental Setup

In this paper, we use the data set of bug reports of Trac Open Source Project and Wordpress which were popular used on the researches of bug report prioritization [23]. Bug reports of Wordpress and Trac collected between *June*, 2004 and *March*, 2013 and *August*, 2003 and *July*, 2013 were used in the research. All the bug reports have been assigned 5 classes of priorities by the triagers correctly such that precision and recall of classification algorithms can be evaluated. After preprocessing the number of features for Trac and Wordpress is $18,231$ and $14,570$. Two best classification algorithms SVM and Naive Bayes for bug report prioritization studied by Lamkanfi [4] were used on the experiments.

Project	#Bug reports	#Reporter	#Versions
Wordpress	23,848	6,013	18
Trac	10,416	4,701	11

Table 1: Data set description

6 Evaluation

In order to evaluate our approach, feature selection were performed before classification, then the precision and recall, popular evaluation criteria for bug report prioritization techniques [12], were calculated for the classification algorithms. A higher precision and recall mean that the specific feature selection techniques are more capable in finding priority relevant features. Precision is the fraction of all the predicted items that are relevant items. For a priority class C_i predicted items are all the bug reports to which C_i is assigned by the classifier. Relevant items are bug reports to which C_i should be assigned.

$$Precision = \frac{|Relevant\ items \cap Predicted\ items|}{|Predicted\ items|} \quad (6)$$

Recall is the fraction of all the relevant items that are predicted items.

$$Recall = \frac{|Relevant\ items \cap Predicted\ items|}{|Relevant\ items|} \quad (7)$$

For each priority class the precision and recall are calculated and then the average values are summarized over all priority classes. As prioritizing bug reports manually took much longer time than classification based bug report prioritization techniques did, existing researches focused on the effectiveness and did not study the efficiency and cost of classification based techniques [12].

7 Experimental Results

	Precision	Recall	Feature selection
Original	0.613	0.565	-
Rank	**0.686**	**0.679**	CO
	0.715	**0.693**	GR
	0.705	**0.654**	IG
	0.632	0.577	OR
	0.623	0.578	RF
	0.630	0.587	SU
Best First	0.654	0.54	CFS
Greedy Stepwise	0.652	0.536	CFS

Table 2: Experimental results of Naive Bayes for Trac project

	Precision	Recall	Feature selection
Original	0.572	0.592	-
Rank	**0.685**	**0.682**	CO
	0.723	**0.719**	GR
	0.618	**0.708**	IG
	0.595	0.603	OR
	0.606	0.637	RF
	0.607	0.613	SU
Best First	0.594	0.644	CFS
Greedy Stepwise	0.592	0.645	CFS

Table 3: Experimental results of Naive Bayes for Wordpress project

In order to evaluate whether feature selection can find priority relevant features, classification were performed on the original feature vector set and the set which processed by feature selection techniques. In order to get more precise results, all the experiments were conducted 30 times and the average value were recorded.

Table 2 and 3 show the experimental results of Trac project of Naive Bayes. The first column shows the search methods. Original means that original feature vector set without feature selection performed on were used for classification. Rank means that features were scored independently. Best First and Greedy Stepwise are two greedy al-

gorithms based searching methods used by CFS. The experimental results show that feature selection techniques CO, GR and IG can get higher precision and recall than the original case and the other feature selection techniques.

	Precision	Recall	Feature selection
Original	0.523	0.517	-
Rank	**0.674**	**0.689**	CO
	0.694	**0.713**	GR
	0.683	**0.668**	IG
	0.545	0.5147	OR
	0.543	0.519	RF
	0.637	0.621	SU
Best First	0.544	0.582	CFS
Greedy Stepwise	0.542	0.536	CFS

Table 4: Experimental results of SVM for Trac project

	Precision	Recall	Feature selection
Original	0.533	0.545	-
Rank	0.557	**0.672**	CO
	0.642	**0.649**	GR
	0.557	0.593	IG
	0.558	0.521	OR
	0.563	0.562	RF
	0.521	0.58	SU
Best First	0.554	0.518	CFS
Greedy Stepwise	0.552	0.526	CFS

Table 5: Experimental results of SVM for Wordpress project

Table 4 and 5 show the results of SVM for two projects. From the two tables we can get similar observations with the results of Naive Bayes. By comparing 4 tables we can see that Naive Bayes performs better than SVM.

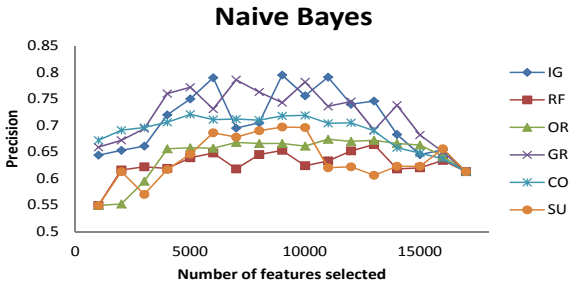

Figure 2: The influence of the number of features on precision for Trac project

Different number of features were set and the specific precision and recall were recorded for each filter method. Figure 2 shows that the precision first rises and then declines with the increase of number of features selected. The precision of GR reaches its peak when the number of features reaches about one third of the total features. This phenomenon gives us suggestions that a subset of priority relevant features are able to give enough information about the priority. We can also see that GR, IG and CO perform better than other techniques.

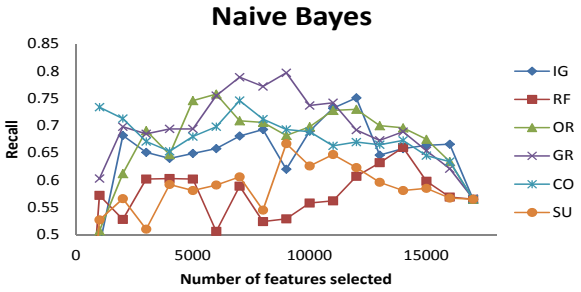

Figure 3: The influence of the number of features on recall for Trac project

In Figure 3 we can see that the peak of recall of GR occurs when the number of features reaches about half of all the features, which is later than those in Figure 2. We can also see that GR is obviously better than RF and SU.

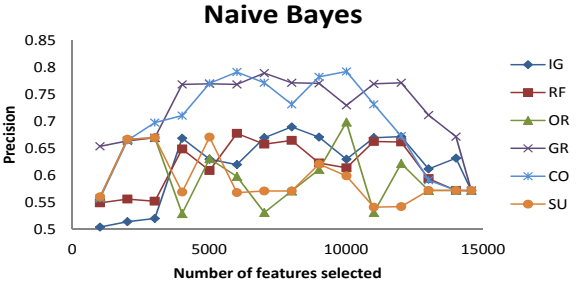

Figure 4: The effects of the number of features on precision for Wordpress project

Figure 4 and 5 show that the peak occurs when the number of features reaches about one third of all the features. Then the peak continues until about $13,000$ features selected. In these two figures CO and GR are still better than others. The performance of IG is similar with RF worse than GR. The reason is that IG dose not take the number of kinds of values of features into consideration. Features that has too many kinds of values that are not relevant with priority are also selected to make IG not as good as GR sometimes.

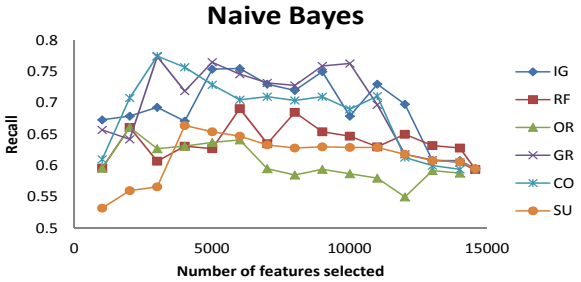

Figure 5: The influence of the number of features on recall for Wordpress project

We also used spearman correlation (SC) to measure the

correlation between the assigned priorities and the actual priorities. A higher value indicates a higher correlation. Table 6 shows the average SC of different feature selection techniques.

	CO	GR	IG	OR	RF	SU	CFS
SC	0.51	0.55	0.53	0.42	0.38	0.38	0.34

Table 6: Spearman correlation on different feature selection techniques

8 Threats to Validity

In our research the threats to validity mainly came from the experimental errors. The priority of bug reports were assigned by humans. They assigned priorities in a subjective way. Different people may have different opinions on the priorities. In the experiment we only used two projects. However, the projects were all large open source projects with large number of bug reports. This made the experiment results more universal. In the future we will study more projects and more bug reports.

9 Conclusion

In this paper feature selection methods are introduced to improve the effect of bug report prioritization using classification models. The experimental results show that feature selection can pick out relevant features and improve the effect of bug report prioritization on two models, i.e., SVM and Naive Bayes. For the feature selection techniques we studied in this paper, IG and GR based on Information Gain and CO based on Pearson Correlation obtained better performance than other feature selection techniques. We also studied the influence of the number of selected features and found that one third to half of the features were enough to get high precision and recall.

References

[1] J. Xuan, H. Jiang, Y. Hu, Z. Ren, W. Zou, Z. Luo, and X. Wu, "Towards effective bug triage with software data reduction techniques," *IEEE transactions on knowledge and data engineering*, vol. 27, no. 1, pp. 264–280, 2015.

[2] Y. Tian, D. Lo, and C. Sun, "Drone: Predicting priority of reported bugs by multi-factor analysis." in *ICSM*, 2013, pp. 200–209.

[3] T. Kremenek and D. Engler, "Z-ranking: Using statistical analysis to counter the impact of static analysis approximations," in *International Static Analysis Symposium*. Springer, 2003, pp. 295–315.

[4] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug," in *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*. IEEE, 2011, pp. 249–258.

[5] J. Novakovic, "The impact of feature selection on the accuracy of naïve bayes classifier," in *18th Telecommunications forum TELFOR*, vol. 2, 2010, pp. 1113–1116.

[6] A. G. Karegowda, A. Manjunath, and M. Jayaram, "Comparative study of attribute selection using gain ratio and correlation based feature selection," *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 271–277, 2010.

[7] M. D'Ambros, M. Lanza, and M. Pinzger, "" a bug's life" visualizing a bug database," in *Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on*. IEEE, 2007, pp. 113–120.

[8] J. Xie, M. Zhou, and A. Mockus, "Impact of triage: a study of mozilla and gnome," in *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, 2013, pp. 247–250.

[9] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*. IEEE, 2008, pp. 346–355.

[10] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 2010, pp. 1–10.

[11] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan, "An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox," in *Reverse Engineering (WCRE), 2011 18th Working Conference on*. IEEE, 2011, pp. 261–270.

[12] J. Kanwal and O. Maqbool, "Bug prioritization to facilitate bug report triage," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 397–412, 2012.

[13] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[14] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, no. 3, pp. 483–519, 2013.

[15] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[16] M. R. Wijaya, R. Saptono, and A. Doewes, "The effect of best first and spreadsubsample on selection of a feature wrapper with naïve bayes classifier for the classification of the ratio of inpatients," *Scientific Journal of Informatics*, vol. 3, no. 2, pp. 41–50, 2016.

[17] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[18] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994.

[19] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[20] Z. Pang, D. Zhu, D. Chen, L. Li, and Y. Shao, "A computer-aided diagnosis system for dynamic contrast-enhanced mr images based on level set segmentation and relieff feature selection," *Computational and mathematical methods in medicine*, vol. 2015, 2015.

[21] S. Fong, Y. Zhuang, H. Luo, K. Liu, and G. Kim, "Finding significant factors on world ranking of e-governments by feature selection methods over kpis," in *International Conference on Soft Computing in Data Science*. Springer, 2015, pp. 65–73.

[22] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[23] Z. Xu, T. He, W. Zhang, Y. Wang, J. Liu, and Z. Chen, "Exploring the influence of time factor in bug report prioritization."