Ryu D, Jang JI, Baik J. A hybrid instance selection using nearest-neighbor for cross-project defect prediction. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(5): 969–980 Sept. 2015. DOI 10.1007/s11390-015-1575-5

A Hybrid Instance Selection Using Nearest-Neighbor for Cross-Project Defect Prediction

Duksan Ryu, Jong-In Jang, and Jongmoon Baik, *Member, ACM, IEEE*

School of Computing, Korea Advanced Institute of Science and Technology, Yuseong-gu, Daejeon 305-701, Korea

E-mail: {dsryu, forestar0719, jbaik}@kaist.ac.kr

Received March 20, 2015; revised July 7, 2015.

Abstract Software defect prediction (SDP) is an active research field in software engineering to identify defect-prone modules. Thanks to SDP, limited testing resources can be effectively allocated to defect-prone modules. Although SDP requires sufficient local data within a company, there are cases where local data are not available, e.g., pilot projects. Companies without local data can employ cross-project defect prediction (CPDP) using external data to build classifiers. The major challenge of CPDP is different distributions between training and test data. To tackle this, instances of source data similar to target data are selected to build classifiers. Software datasets have a class imbalance problem meaning the ratio of defective class to clean class is far low. It usually lowers the performance of classifiers. We propose a Hybrid Instance Selection Using Nearest-Neighbor (HISNN) method that performs a hybrid classification selectively learning local knowledge (via k-nearest neighbor) and global knowledge (via naïve Bayes). Instances having strong local knowledge are identified via nearest-neighbors with the same class label. Previous studies showed low PD (probability of detection) or high PF (probability of false alarm) which is impractical to use. The experimental results show that HISNN produces high overall performance as well as high PD and low PF.

Keywords software defect analysis, instance-based learning, nearest-neighbor algorithm, data cleaning

1 Introduction

It is crucial to prevent financial and human losses due to the failures of software-intensive systems. Since software defects may cause system failures, they should be detected as soon as possible before release. Software quality assurance resources are mostly limited and thereby need to be allocated cautiously. Software defect prediction (SDP) is one of active research fields in software engineering to identify defect-prone modules in software intensive systems. By directing valuable resources to fault-prone modules, SDP can help manage software quality effectively. SDP is implemented with the help of machine learning algorithms which require sufficient local training data to accurately predict defect-prone modules. It means that sufficient training data should have the same distribution as test data. This case is called within-project defect prediction (WPDP). There are some cases where such local data are not available, e.g., pilot projects within an organization. Additionally, the cost of identifying labels of test data is usually expensive. In these situations, companies can employ cross-project defect prediction (CPDP) using external data to build a classifier. Typically, on software defect datasets, the ratio of the defective class to the clean class is far low. This problem is called "the class imbalance", which lowers the performance of classifiers. To cope with the class imbalance, various approaches, e.g., data resampling[1], cost-sensitive learning[2], and ensemble methods[3], have been suggested. Particularly, in the context of the class imbalance learning, it is significant to produce high performance for the minority class while not severely

Regular Paper

Special Section on Software Systems

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science, ICT and Future Planning (MSIP)) under Grant No. NRF-2013R1A1A2006985 and Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) under Grant No. R0101-15-0144, Development of Autonomous Intelligent Collaboration Framework for Knowledge Bases and Smart Devices.

lowering the performance of the majority class. Both the defect detection rate and the overall performance are measured to evaluate the performance of classifiers since they have a trade-off relationship. For the defect detection rate, the probability of detection (PD) and the probability of false alarm (PF) are used. For the overall performance, the metric utilizing both PD and PF values, e.g., balance[4], is employed. Since it measures how well PD and PF values are balanced, higher value is better.

As Turhan *et al.*[5] pointed out, cross-project (CP) data increases both PD and PF. The reason is that more known sources of defects and more irrelevancies are introduced together to the prediction. Since classifiers producing high PF rates are not practical to use, it is necessary to study the prediction strategy producing acceptable low PF rates.

The main objective of this research is to develop an effective prediction model producing high PD and low PF under cross-project settings. We propose a Hybrid Instance Selection Using Nearest Neighbor (HISNN) method to overcome the problems mentioned above by filtering out irrelevant instances. The proposed approach performs a hybrid classification with consideration of knowledge about both the class boundary and the nearest cluster. To evaluate HISNN, this paper investigates the following research questions:

• RQ1: does HISNN generate performance practical to use compared with other models under cross-project settings?

• RQ2: can HISNN produce the prediction performance comparable to within-project defect prediction?

The prediction performance of HISNN is assessed by comparing it with the existing CPDP and the techniques which are suggested for solving the class imbalance problem. Wilcoxon rank-sum test[6] at a 5% significance level and the A-statistics effect size test[7] are conducted for the evaluation of the results. The results show HISNN is significantly better than classification models under CP settings. Furthermore, HISNN produces overall performance comparable to WPDP. Therefore, HISNN can be used to manage software quality effectively by helping the decision making of resource allocations.

The remainder of this paper is organized as follows. In Section 2, we describe related work. In Section 3, our proposed classification model is explained. The experimental setup is described in Section 4. The results of the experiments are described in Section 5. We explain the threats to validity of our approach in Section 6. In the last section, we conclude the paper and discuss future work.

2 Related Work

2.1 Software Defect Prediction

Software defect prediction (SDP) aims at identifying fault-prone modules in software. Not only do software quality assurance activities like testing and inspection require labor-intensive work, but also time and human resources for such activities are limited. It is necessary to allocate resources optimally for the purpose of improving software quality and reducing the cost of software quality control. Resources can be effectively allocated more to fault-prone modules owing to SDP. Researchers proposed a variety of SDP models[8-13]. In most cases, within-project defect prediction (WPDP) models have been studied. They are useful only if companies collect and manage sufficient local historical data. There are some cases where such local data are not available, e.g., pilot projects within a development organization. Small and mid-size organization may not have enough financial/human resources to collect and manage historical data. Cross-project defect prediction (CPDP) is a defect prediction mechanism where a company without sufficient historical data can employ the data collected from other companies to build a classifier. More and more studies to solve the issues of CPDP have been proposed recently.

Zimmermann *et al.*[14] performed 622 CPDP cases with only 3.4% success. They indicated that the data and process characteristics are key elements for the success of CPDP. They suggested more researchers should study CPDP.

He *et al.*[15] proposed the instance selection based approach for CPDP. According to experimental results, they indicated that distributional characteristics, e.g., mean, variance, are closely related to the prediction results.

Ma *et al.*[16] proposed a transfer learning based CPDP technique called Transfer Naïve Bayes. The similarity weights are calculated based on distributional characteristics and then used for constructing a classifier.

Nam *et al.*[17] applied a transfer learning technique on software defect datasets. They asserted that different normalization options based on distributional characteristics affect the performance of CPDP.

Turhan *et al.*[5] suggested the nearest neighbor-based relevancy filtering approach for CPDP. Instances

from a training dataset were chosen to create a filtered dataset similar to the test set. k-nearest neighbor using Euclidean distance was employed to measure the similarity between training and test instances.

Previous studies of CPDP[5,14-17] utilized distributional characteristics, e.g., the mean, median, maximum and minimum, to calculate the similarity between source and target data. The relevancy filtering method (Burak filter)[5] employs a k-NN method to reduce PF rates compared with the CP model not applying the relevancy filter. In our research, we aim at reducing PF rates further and obtaining optimal overall performance (i.e., high PD and low PF). To show the effectiveness of the proposed approach, it is compared with the relevancy filtering method.

2.2 Class Imbalance Learning

The class imbalance problem indicates that the ratio of defective instances to non-defective instances is much low[18]. Through a systematic literature review on SDP, Hall *et al.*[8] indicated that some classification models might be negatively influenced by the class imbalance. They asserted more SDP studies should take into account the class imbalance issue to improve the prediction accuracy.

Although the number of the minority class is far less than that of the majority, its correct classification is more valuable. The reason is that the misclassification of the minority class is directly related to software quality. If fault-prone modules are not correctly predicted, they would not have a chance to be inspected or tested by limited software quality assurance control. Thus, the class imbalance learning aims at obtaining the classifier producing high overall performance (e.g., balance) as well as high performance for the minority without lowering the performance of the majority class.

There have been various approaches to identifying the minority class by taking into account the key characteristics of the class imbalance at algorithm and data levels. A simple way to deal with the class imbalance at the data level is sampling methods including undersampling and over-sampling. At the algorithm level, cost-sensitive learning techniques are widely applied. Different costs of misclassification are assigned to the training instances.

Grbac *et al.*[19] studied the relationships between the class imbalance of defect datasets and the performance stability of machine learning models. They asserted that the prediction ability of classification models can be negatively influenced by a high level of imbalance.

Raman and Ioerger[20] proposed the learning algorithm using search ring (LASER) framework to deal with class imbalance. They proposed a method which adopts the instance selection strategy, i.e., selective learning. In highly imbalanced domains, the minority class instances will be incorrectly classified if global information is used. Consequently, they employed different instance selection strategies at different regions of instance space.

In our study, we adopt the concept of LASER framework to produce high prediction performance under cross-project settings. Based on the work of Beyer *et al.*[21], our approach additionally considers the nearest cluster to improve the performance of nearest neighbor.

3 HISNN Approach

Under CP settings, it is required to address different distributions between source and target data. To this end, we propose a Hybrid Instance Selection Using Nearest Neighbor (HISNN) approach, a selective learning technique based on the local knowledge of source data. It consists of two main phases, i.e., test data instance selection and training data instance filtering. Fig.1 shows the overall defect prediction process using the HISNN method.

3.1 Testing Data Instance Selection

Our proposed HISNN extends LASER[20] to provide the high prediction performance for CPDP. The training instances similar to the test instance are identified. Hamming distance is used as a similarity measure. The Min-Ham distance is the minimum distance in Hamming space where there is a neighbor. Suppose $Min\text{-}Ham(\boldsymbol{x}_i) = \varepsilon$, $HammingDistance(\boldsymbol{x}_i, \boldsymbol{x}_j) \geqslant \varepsilon$, $\forall \boldsymbol{x}_i \in \boldsymbol{X}$, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are element vectors of $\boldsymbol{X}$ (instance space).

With the target instance as the center, a search ring of Min-Ham radius in Hamming space is formed. There are three possible cases when choosing all instances on the ring.

- In case 1, only one instance is present (at Min-Ham distance).
- In case 2, all instances are of the same class.
- In case 3, selected instances are of the different classes.

In case 1, a test instance is closer to a single training instance than all the others. In LASER, an instance

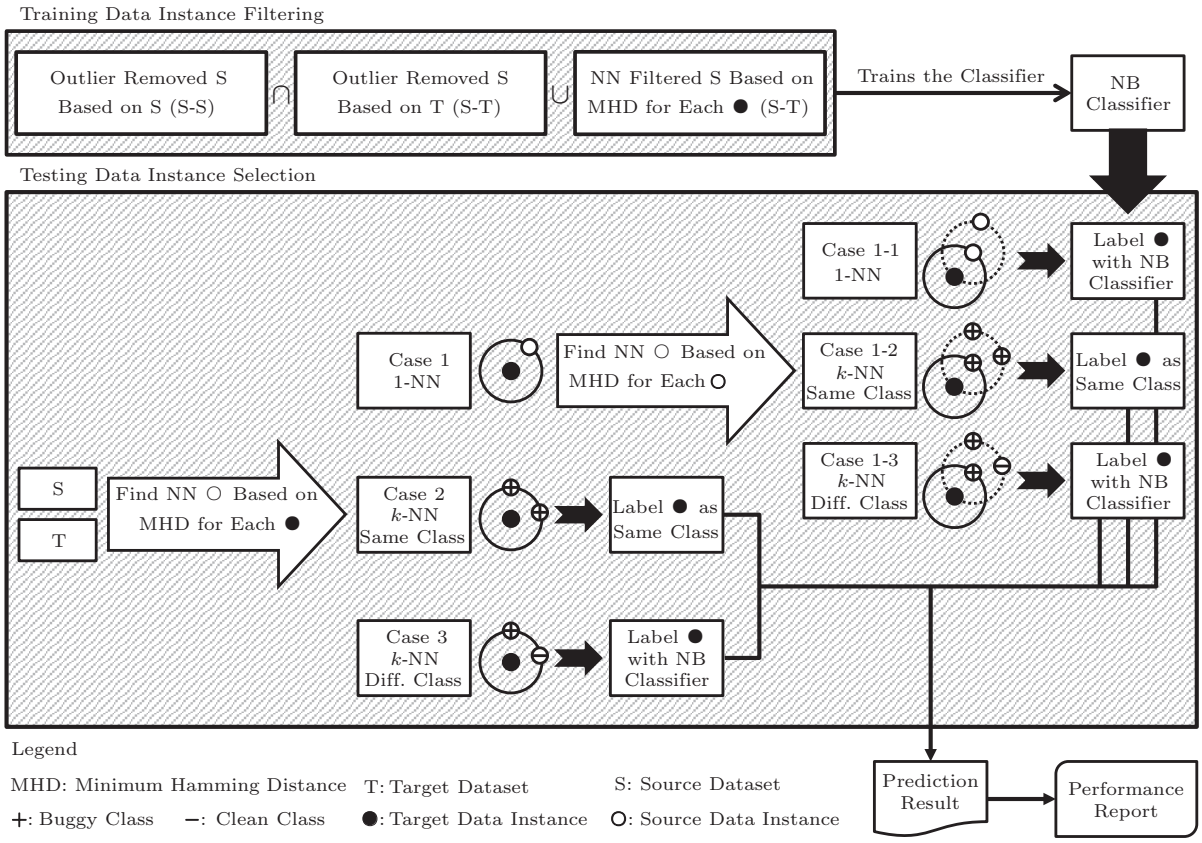

Fig.1. Overall process with hybrid instance selection using nearest neighbor approach.

in case 1 returns the class of the search ring like 1-NN. Unlike LASER, our approach further explores the nearest cluster of the close single training instance having weak local knowledge. Beyer *et al.*[21] pointed out that the nearest cluster should be considered when k-NN is used as a predictor.

For case 1, additional three cases (i.e., case 1-1, case 1-2, case 1-3) are possible. With the single training instance as the center, a search ring of Min-Ham radius in Hamming space is formed.

• In case 1-1, only one instance is present (at Min-Ham distance).

• In case 1-2, all instances are of the same class.

• In case 1-3, selected instances are of the different classes.

Since an instance in case 1-1 has weak local knowledge and instances in case 1-3 fall near the decision boundary, they need the global probability distribution to determine the class label. Thus, for the instances in case 1-1 and case 1-3, the naïve Bayes learner is used. An instance in case 1-2 has strong local knowledge. Thus, the instance in case 1-2 returns the class of the search ring like k-NN in Hamming space. Likewise, an instance in case 2 returns the class of the search ring. For instances in case 3, the naïve Bayes learner is employed.

HISNN employs a hybrid instance selection strategy, i.e., k-closest instances for cases 1-2 and 2 and all instances for cases 1-1, 1-3 and 3. The reason of doing a local instance selection based on the local knowledge of source data is that more extraneous instances might be included by the inclusion of other instances. An instance having the weak local knowledge falls near the decision boundary and thus requires the global probability distribution to identify its class. Through the hybrid instance selection strategy, the strong local knowledge is explicitly used to produce the high prediction performance.

3.2 Training Data Instance Filtering

HISNN approach utilizes outlier removing technique and nearest neighbor approach to filter out the source data instances that might hinder the prediction performance.

First of all, the outliers among the source project data are identified and removed. In the HISNN approach, the outliers can be detected based on the Mahalanobis distance[22]. The Mahalanobis distance measures the distance between a data instance and the whole data distribution. Its idea is to measure how many standard deviations away the data instance is from the mean of the distribution. We have considered the data instance as an outlier if it is more than about three standard deviations away from the mean of the distribution. The HISNN approach filters the source instances out with this scheme based on both the source dataset distribution and the target dataset distribution. The intuition behind this process is quite straight forward. The outliers from the source dataset distribution are regarded to be the noise of the source project data and the outliers from the target dataset distribution are obviously the irrelevant data instances that do not aid the prediction. After this process, we obtain the filtered source dataset containing the instances without outliers.

For better CPDP, source instances similar to the target data should be identified. Thus, the HISNN approach performs the nearest neighbor filtering based on a search ring of Min-Ham radius around each test instance. Only the source instances inside each search ring will remain in the source dataset.

Finally, the HISNN approach merges the source instances without outliers and the source instances similar to the test data to get the final training data. The test instances which need the global probability distribution to determine the class label are classified by the naïve Bayes using this merged training data.

3.3 Performance Report

Typically, when considering the class imbalance problem, a classifier is assessed by both the individual and the overall performance. The individual performance on the defective class is evaluated by the probability of detection (PD) and the probability of false alarm (PF). The overall performance is assessed by balance which uses both PD and PF values. Table 1 shows the confusion matrix used for performance evaluation.

Balance is defined as: $balance = 1 - \frac{\sqrt{(0-PF)^2+(1-PD)^2}}{\sqrt{2}}$. It measures a Euclidean distance between the real (PD, PF) point and the ideal (PD, PF) point, i.e., (1, 0). Balance shows how well the performance between the defective and the non-defective classes is balanced by a classifier. PD and balance are desired to be high in contrast to PF.

Table 1. Confusion Matrix

		Predicted Class	
		Buggy	Clean
Actual class	Buggy	TP (true positive)	FN (false negative)
	Clean	FP (false positive)	TN (true negative)

4 Experimental Setup

To show the performance of the proposed approach, we set up the comparative experiments. For RQ1, LASER[20], naïve Bayes and Burak filter[5] under CP settings are compared with our approach. For RQ2, naïve Bayes under WP settings is compared with HISNN.

The core concept of LASER is adopted into HISNN for CPDP. LASER is chosen for experiments to check whether it can be solely used for CPDP. Naïve Bayes is chosen because it generally shows high performance in SDP compared with other predictors[8]. Burak filter is selected since it is widely employed in CPDP studies[4-5,23].

To evaluate the performance, our approach is compared with other classifiers on NASA datasets. PD, PF and balance are selected as performance measures since they are considered useful in the context of the class imbalance.

Hypotheses to answer RQ1 and RQ2 are formulated as follows:

- $H1_0$: HISNN is not statistically better than the other approaches under cross-project settings;
- $H1_A$: HISNN is statistically better than the other approaches under cross-project settings;
- $H2_0$: HISNN is neither statistically better than nor similar to within-project defect prediction;
- $H2_A$: HISNN is either statistically better than or similar to within-project defect prediction.

4.1 NASA Datasets

NASA datasets obtained from PROMISE repository① and NASA MDP Software Defect Datasets② have been widely employed for the case studies of SDP. Table 2 describes the characteristics of

①http://openscience.us/repo/, July 2015.

②http://nasa-softwaredefectdatasets.wikispaces.com/, July 2015.

NASA datasets. Table 3 shows the features of NASA datasets used in the experiments. To organize the cross-project settings, each NASA dataset is selected to be test data and the remaining datasets are utilized as training data. For WPDP, the 50 : 50 random split is used to obtain training and test data. Each CP and WP case was run 100 times iteratively and the median performance is reported.

Table 2. Projects of NASA Datasets

Project	Number of Instances	Percentage of Buggy	Description
cm1	498	9.83	Spacecraft instrument
kc1	1 183	24.85	Storage management
kc2	522	20.49	Science data processing
kc3	458	9.38	Storage management
mc2	161	32.30	Video guidance software
mw1	403	7.69	A zero-gravity experiment
pc1	1 109	6.94	Flight software

Table 3. Features of NASA Datasets

Type	Feature
LOC	Code and comment loc, comment loc, executable loc, total loc
Halstead	Halstead difficulty, Halstead effort, Halstead error, Halstead length, Halstead time, Halstead volume, unique operands, unique operators, total operands, total operators
McCabe	Cyclomatic complexity, design complexity
Branch	Branch count

4.2 Classification Models

We employ k-NN from MATLAB③ and naïve Bayes from the WEKA machine learning toolkit[24]. As Turhan *et al.*[5] recommended, we apply log-filter to all numeric values (i.e., replacing N with $\ln(N)$) only when naïve Bayes is used. Log-filter is not used when k-NN is applied.

5 Experimental Results

Table 4 shows the median PD, PF and balance performance of classification models. The best result over each experiment is marked in boldface.

Fig.2 illustrates scatter plot of median PD and PF values of five models over seven datasets. The better classification model has more points at the bottom right of the area since high PD and low PF are desired. LASER shows the lowest PF value (0.245). However, its PD value (0.534) is also the lowest. In Fig.2, all points that represent LASER's performance are located at the bottom left of the area. Since the low PD value indicates that the defect detection rate is low, this case is impractical to use.

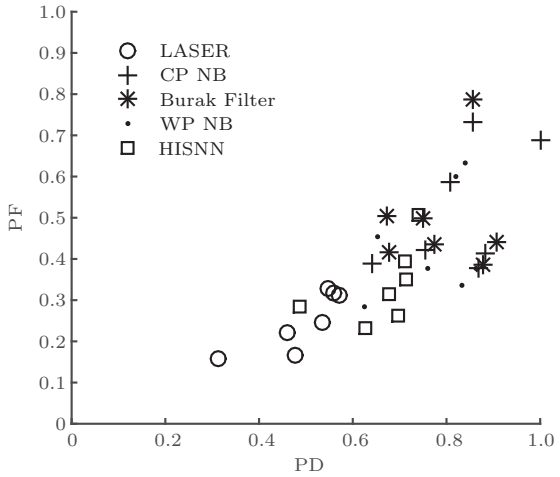

Fig.2. Scatter plot of median PD and PF values of five models over seven datasets.

Table 4. Median PD, PF, and Balance Performance of Classification Models

Target	LASER			CP Naïve Bayes			Burak Filter			WP Naïve Bayes			HISNN		
Data	PD	PF	Bal	PD	PF	Bal	PD	PF	Bal	PD	PF	Bal	PD	PF	Bal
cm1	0.571	**0.311**	0.325	**0.755**	0.423	0.654	**0.775**	0.436	0.652	0.760	0.377	**0.683**	0.714	0.351	0.679
kc1	0.312	**0.159**	0.501	0.642	0.390	**0.625**	0.751	0.499	0.605	**0.840**	0.633	0.541	0.486	0.283	0.585
kc2	0.476	**0.166**	0.611	0.869	0.378	0.716	**0.878**	0.387	0.712	0.833	0.336	**0.730**	0.626	0.231	0.689
kc3	0.534	**0.245**	0.628	0.883	0.414	0.695	**0.906**	0.440	0.681	0.863	0.377	0.713	0.697	0.262	**0.716**
mc2	0.461	**0.220**	0.588	**0.807**	0.587	0.563	0.673	0.504	0.574	0.653	0.454	0.592	0.711	0.394	**0.654**
mw1	0.548	0.327	0.605	**1.000**	0.688	0.513	0.677	0.416	0.627	0.625	**0.284**	0.671	0.677	0.314	**0.681**
pc1	0.558	**0.318**	**0.614**	**0.857**	0.732	0.472	**0.857**	0.788	0.433	0.820	0.600	0.553	0.740	0.507	0.596
Med	0.534	**0.245**	0.611	**0.857**	0.423	0.625	0.775	0.440	0.627	0.820	0.377	0.671	0.697	0.314	**0.679**

Note: PD: probability of detection, PF: probability of false alarm, Bal: balance, Med: median.

③http://www.mathworks.com, July 2015.

Naïve Bayes under CP settings (CP NB) produces the highest PD (0.857). However, it produces high PF (0.423) as well. In contrast with LASER, CP NB shows more points at the top right of the area. When practitioners use defect predictors with high false alarm rates, a great deal of time and budget must be allocated to explore erroneous alarms. In cases of mw1 and pc1, PF values show 0.688 and 0.732 respectively. As Turhan *et al.*[5] pointed out, false alarm rates as high as 64% could not be accepted by user groups. For most software applications, it is impractical to use the predictors producing very high PF rates.

To deal with such problems, Burak filter was proposed by Turhan *et al.*[5] There are several cases of reducing PF values, e.g., from 0.688 to 0.416. However, there is still a case having very high PF value (0.788). Like CP NB, Burak filter shows more points at the top right of the area in Fig.2.

Naïve Bayes under WP settings (WP NB) shows high PD (0.820) and low PF (0.377) as expected. However, there are cases producing high PF values (0.633, 0.454, and 0.600).

Our proposed HISNN shows the highest overall performance (0.679) compared with the other models. HISNN provides high accuracy for the minority class without severely lowering the accuracy of the majority class which is important in the context of the class imbalance.

The predictive results are analyzed on the basis of the research of [10, 25] that assesses the variability of the classification models across multiple runs. We use the mini boxplot to display the first, the second and the third quartile of each case. The predictors are sorted by their medians. Visually, a bar indicates the first-third quartile range and a circle represents the median. The minimum and the maximum values are not shown. The 100 data points for each target project are merged for each measure. The first, the second and the third quartile are calculated with 700 data points for seven projects. Fig.3 shows mini boxplots of median PD, PF and balance of five models for all the projects sorted by median. We show the predictive results for individual projects in Appendix. We conduct the Wilcoxon rank-sum test[6] at a 5% significance level to check which classifiers are better than others. To assess the magnitude of the improvement, we perform A-statistics effect size test[7]. An A-statistic of greater than 0.64 (or less than 0.36) means a medium effect size. Seven hundred data points for seven projects are used for the two tests. Table 5 shows the comparison of HISNN with classifiers based on the two tests. The boldface in the table shows the significantly better result of HISNN with p-value < 0.05 or A-statistics > 0.64 for PD and balance (for PF, A-statistics < 0.36).

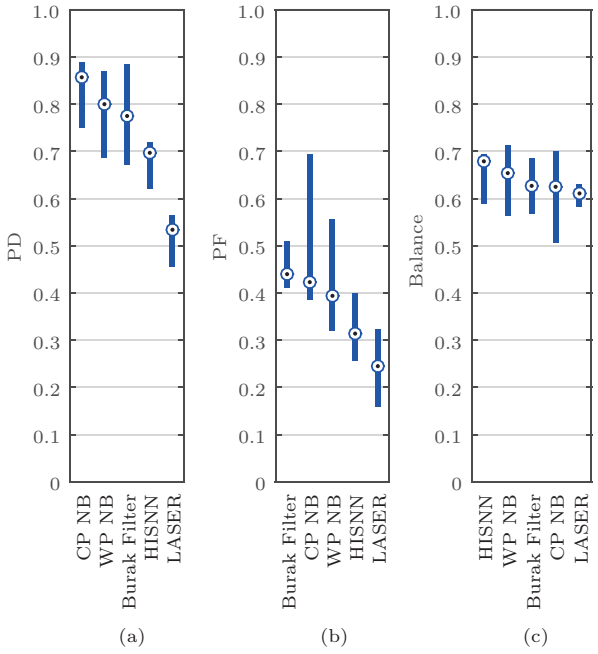

Fig.3. Mini boxplots of (a) median PD, (b) PF, and (c) balance values of five models over seven datasets.

Table 5. Comparison of HISNN with Other Classification Models

		LASER	CP NB	Burak Filter	WP NB
PD	p-value	$<<$ **0.001**	$<<$ 0.001	$<<$ 0.001	$<<$ 0.001
	A-statistic	**0.910**	0.100	0.190	0.200
PF	p-value	$<<$ 0.001	$<<$ **0.001**	$<<$ **0.001**	$<<$ **0.001**
	A-statistic	0.750	**0.120**	**0.140**	**0.270**
Bal	p-value	$<<$ **0.001**	$<<$ **0.001**	$<<$ **0.001**	**0.009**
	A-statistic	**0.770**	**0.650**	**0.660**	0.530

The experimental results of HISNN can be analyzed based on the two research questions.

For RQ1, in order for a classifier to be practical to use, it should provide high overall performance as well as high PD and low PF. In Table 5, the Wilcoxon rank-sum test shows that the differences in balance values between HISNN and other classifiers under CP settings are statistically significant with p-value $<< 0.001$. The effect size for balance is higher than 0.64 (LASER: 0.77, CP NB: 0.65, Burak Filter: 0.66), meaning the

medium effect. Thus, we reject $H1_0$. The results represent HISNN can be practically used compared with other classifiers under CP settings.

For RQ2, we check if HISNN is comparable to WPDP. In Table 5, the Wilcoxon rank-sum test shows the difference in balance values between HISNN and naïve Bayes under WP settings is statistically significant with p-value $<< 0.05$. In terms of the effect size test for balance, the effect size is 0.53 indicating a small effect. Since HISNN is comparable to WPDP, we reject $H2_0$.

The benefit of HISNN is to obtain acceptably low PF rates as well as high overall performance. However, the weakness of HISNN is that PD rates are somehow low compared with those of WPDP. The possible reason of the outcome is as follows. Under CP settings, it is crucial to select instances of source project similar to target project. In HISNN, based on the local knowledge of source data, two cases (a local search and a global search) are separated. For each case, more irrelevant instances might not be included. Consequently, low PF rates could be obtained.

6 Threats to Validity

6.1 Construct Validity

Minimum hamming distance is employed as a similarity measure among training instances in our approach. This distance metric might not be able to capture the true feature distributional characteristics.

6.2 External Validity

NASA datasets used for validation are open source software project data. If our proposed approach is built on closed software projects developed under different environments, it might produce better/worse performance.

6.3 Statistical Conclusion Validity

In this study, Wilcoxon rank-sum test is performed. It is the recommended approach to check the statistical significant difference between two classifiers' performances.

7 Conclusions

Since software defect may cause system failures, they should be detected as soon as possible before release. Software defect prediction can help manage software quality effectively by directing quality assurance resources to defective modules in that such resources are limited. In cases where historical data are not available, cross-project defect prediction (CPDP) using external data to build a classifier can be employed. Its poor performance is mainly due to the different distributions between the source and the target projects. One of the typical approaches for CPDP is to identify source instances similar to a target project by using the similarity measure based on distributional characteristics.

On software defect datasets, the ratio of the defective class to the clean class is far low. This problem called the class imbalance lowers performance of specific classification models. To deal with the class imbalance, data resampling, cost-sensitive learning, and ensemble methods have been studied.

In this paper, we proposed a Hybrid Instance Selection Using Nearest Neighbor (HISNN) method using a hybrid classification to address the class imbalance for CPDP. Applicability of the selective learning based on the nearest neighbor via the class boundary identification for CPDP was investigated. To address the different distributional characteristics, training data instances were filtered based on the minimum hamming distance used as the similarity measure. Through Wilcoxon rank-sum test and A-statistics effect size test, we showed that HISNN is better in terms of the overall performance. Through the proposed approach correctly predicting defective modules, the cost of software quality control can be effectively managed.

As future work, we may study if there exist optimal class imbalance learning techniques for CPDP that maximize the PD while minimizing the PF.

References

[1] Gao K, Khoshgoftaar T. Software defect prediction for high-dimensional and class-imbalanced data. In *Proc. the 23rd SEKE*, July 2011, pp.89-94.

[2] Zheng J. Cost-sensitive boosting neural networks for software defect prediction. *Expert Syst. Appl.*, 2010, 37(6): 4537-4543.

[3] Wang S, Yao X. Using class imbalance learning for software defect prediction. *IEEE Trans. Reliab.*, 2013, 62(2): 434-443.

[4] Turhan B, Tosun Mısırlı A, Bener A. Empirical evaluation of the effects of mixed project data on learning defect predictors. *Inf. Softw. Technol.*, 2013, 55(6): 1101-1118.

[5] Turhan B, Menzies T, Bener A B, Di Stefano J. On the relative value of cross-company and within-company data for defect prediction. *Empir. Softw. Eng.*, 2009, 14(5): 540-578.

[6] Wilcoxon F. Individual comparisons by ranking methods. *Biometrics Bull.*, 1945, 1(6): 80-83.

[7] Vargha A, Delaney H D. A critique and improvement of the "CL" common language effect size statistics of McGraw and Wong. *J. Educ. Behav. Stat.*, 2000, 25(2): 101-132.

[8] Hall T, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans. Softw. Eng.*, 2012, 38(6): 1276-1304.

[9] Arisholm E, Briand L C, Johannessen E B. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *J. Syst. Softw.*, 2010, 83(1): 2-17.

[10] D'Ambros M, Lanza M, Robbes R. Evaluating defect prediction approaches: A benchmark and an extensive comparison. *Empir. Softw. Eng.*, 2012, 17(4/5): 531-577.

[11] Dejaeger K, Verbraker T, Basesens B. Toward comprehensible software fault prediction models using Bayesian network classifiers. *IEEE Trans. Softw. Eng.*, 2013, 39(2): 237-257.

[12] Elish K O, Elish M O. Predicting defect-prone software modules using support vector machines. *J. Syst. Softw.*, 2008, 81(5): 649-660.

[13] Singh Y, Kaur A, Malhotra R. Empirical validation of object-oriented metrics for predicting fault proneness models. *Softw. Qual. J.*, 2009, 18(1): 3-35.

[14] Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. In *Proc. the 7th ESEC/FSE*, August 2009, pp.91-100.

[15] He Z, Shu F, Yang Y, Li M, Wang Q. An investigation on the feasibility of cross-project defect prediction. *Autom. Softw. Eng.*, 2011, 19(2): 167-199.

[16] Ma Y, Luo G, Zeng X, Chen A. Transfer learning for cross-company software defect prediction. *Inf. Softw. Technol.*, 2012, 54(3): 248-256.

[17] Nam J, Pan S J, Kim S. Transfer defect learning. In *Proc. the 35th Int. Conf. Softw. Eng.*, May 2013, pp.382-391.

[18] Tan P N, Steinbach M, Kumar V. Introduction to Data Mining. Addison Wesley, 2006.

[19] Grbac T, Mausa G, Bašić B. Stability of software defect prediction in relation to levels of data imbalance. In *Proc. the 2nd SQAMIA*, Sept. 2013, pp.1:1-1:10.

[20] Raman B, Ioerger T R. Enhancing learning using feature and example selection. Technical Report, Department of Computer Science, Texas A&M Univ., 2003.

[21] Beyer K, Goldstein J, Ramakrishnan R, Shaft U. When is "nearest neighbor" meaningful? In *Lecture Notes in Computer Science 1540*, Beeri C, Buneman P (eds.), Springer-Verlag, 1999, pp.217-235.

[22] Mahalanobis P C. On the generalised distance in statistics. *Proc. Natl. Inst. Sci.*, 1936, 2(1): 49-55.

[23] Turhan B, Tosun A, Bener A. Empirical evaluation of mixed-project defect prediction models. In *Proc. the 37th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, Aug. 30-Sept. 2, 2011, pp.396-403.

[24] Hall M, Frank E, Holmes G *et al.* The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.*, 2009, 11(1): 10-18.

[25] Menzies T, Milton Z, Turhan B, Cukic B, Jiang Y, Bener A. Defect prediction from static code features: Current results, limitations, new approaches. *Autom. Softw. Eng.*, 2010, 17(4): 375-407.

Duksan Ryu earned his Bachelor's degree in computer science from Hanyang University, Seoul, in 1999, and Master's dual degree in software engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, and Carnegie Mellon University, Pittsburgh, in 2012. He is a Ph.D. student in the School of Computing at KAIST. His research areas are software defect prediction and software reliability engineering.

Jong-In Jang earned his Bachelor's degree in computer science from the School of Computing, KAIST, in 2014. He is an master student in the School of Computing at KAIST. His research areas are software reliability engineering and requirements engineering.

Jongmoon Baik earned his B.S. degree in computer science and statistics from Chosun University, Gwangju, in 1993. He received his M.S. and Ph.D. degrees in computer science from University of Southern California, Los Angeles, in 1996 and 2000 respectively. He worked as a principal research scientist at Software and Systems Engineering Research Laboratory, Motorola Labs, where he was responsible for leading many software quality improvement initiatives. Currently, he is an associate professor in the School of Computing at KAIST. He is a member of ACM and IEEE. His research activity and interest are focused on software six sigma, software reliability & safety, and software process improvement.

Appendix

In Figs.A1∼A3, mini boxplots of median PD, PF and balance values of five classifiers for each target project over seven datasets are shown. In Tables A1∼A7, the comparison results of HISNN with other classifiers for each target project are shown.

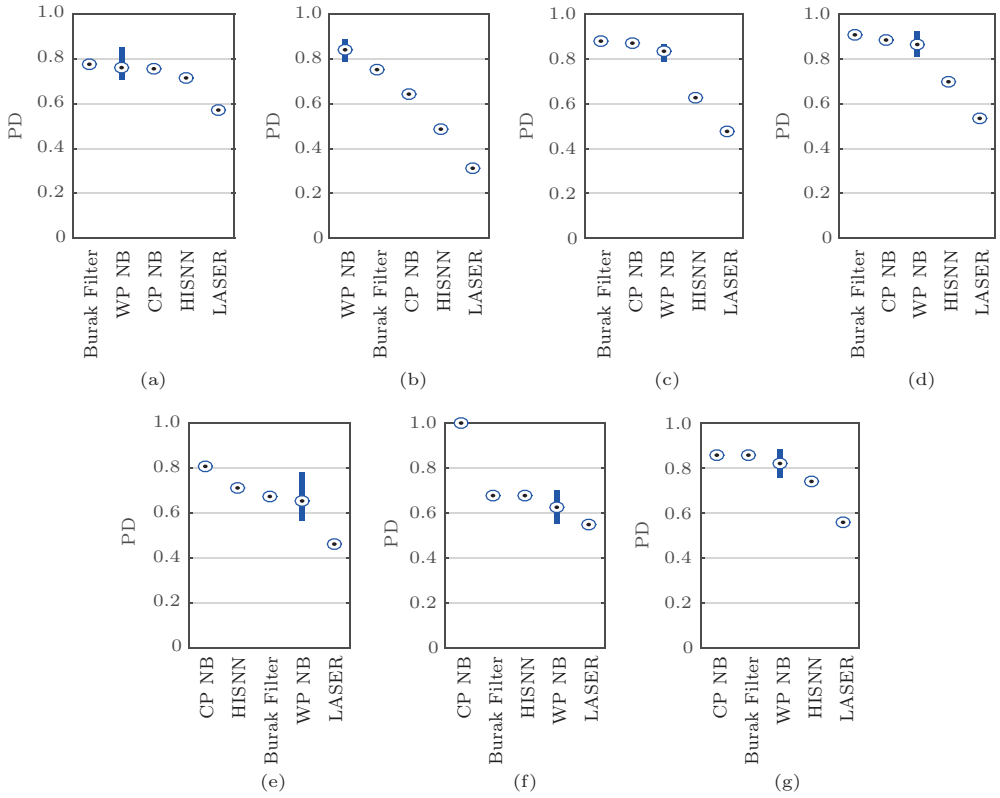

Fig.A1. Mini boxplots of median PD values of five models over seven datasets. (a) cm1. (b) kc1. (c) kc2. (d) kc3. (e) mc2. (f) mw1. (g) pc1.

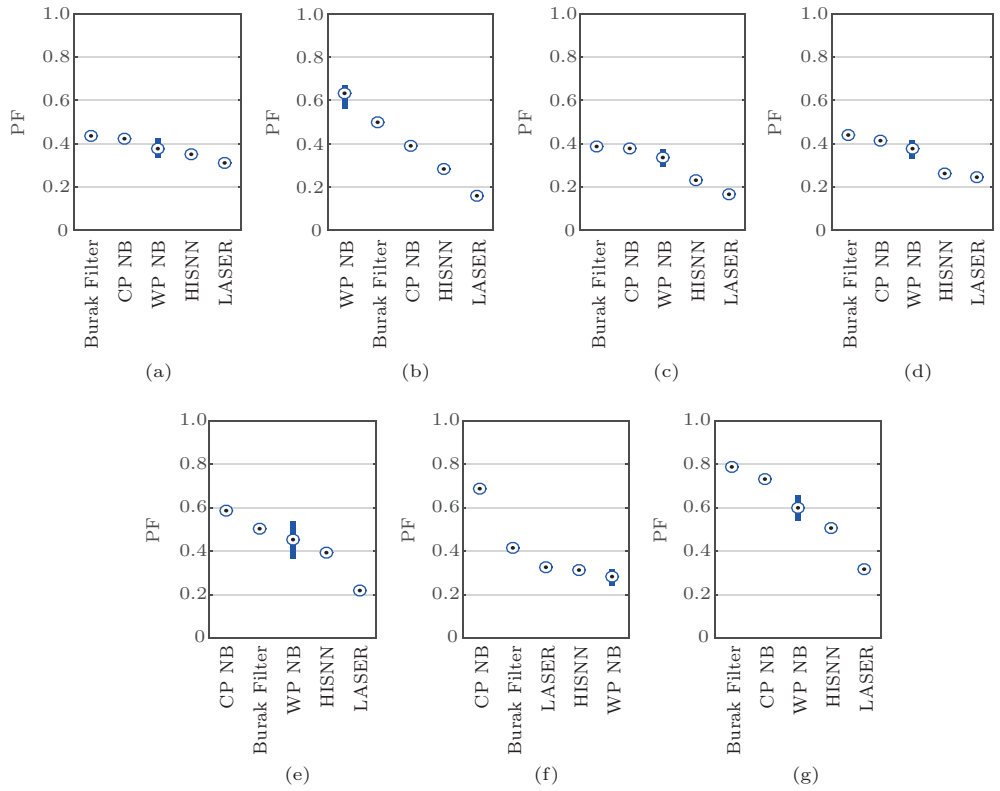

Fig.A2. Mini boxplots of median PF values of five models over seven datasets. (a) cm1. (b) kc1. (c) kc2. (d) kc3. (e) mc2. (f) mw1. (g) pc1.

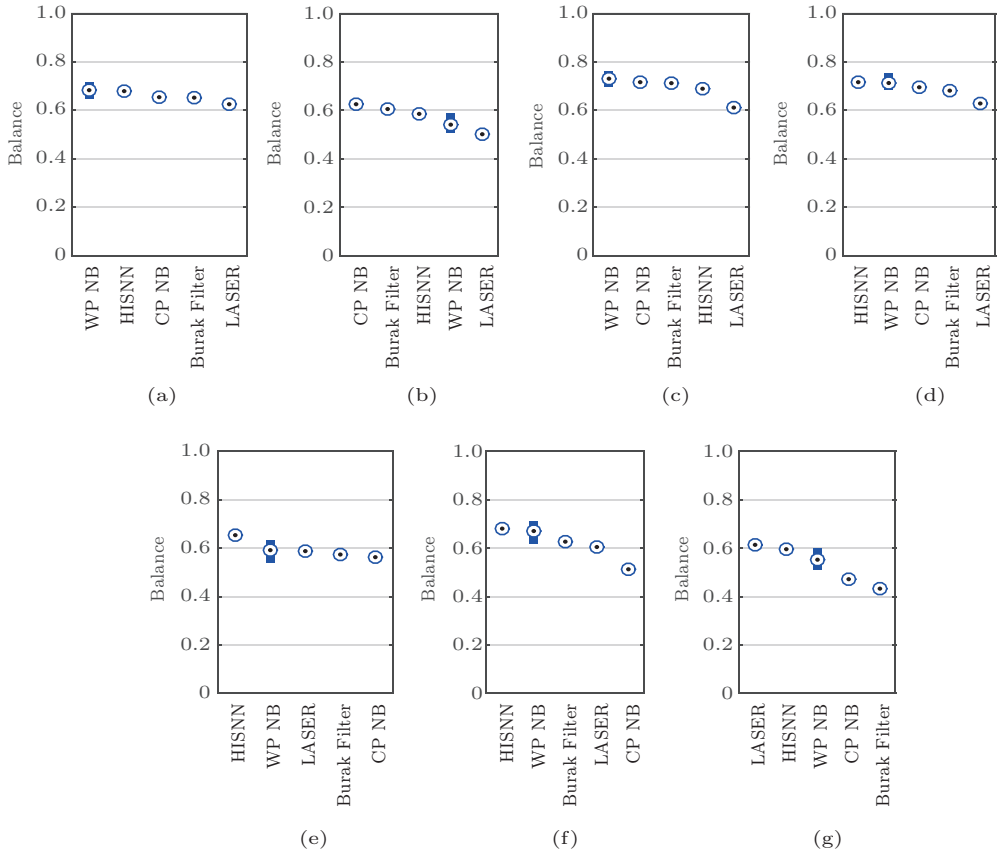

Fig.A3. Mini boxplots of median balance values of five models over seven datasets. (a) cm1. (b) kc1. (c) kc2. (d) kc3. (e) mc2. (f) mw1. (g) pc1.

Table A1. Comparison of HISNN with Other Classification Models for the cm1 Project

		LASER	CP NB	Burak Filter	WP NB
PD	*p*-value	**<<0.001**	<<0.001	<<0.001	<<0.001
	A-statistic	**1.000**	0.000	0.000	0.220
PF	*p*-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	1.000	**0.000**	**0.000**	**0.270**
Bal	*p*-value	**<<0.001**	**<<0.001**	**<<0.001**	0.035
	A-statistic	**1.000**	**1.000**	**1.000**	0.420

Table A2. Comparison of HISNN with Other Classification Models for the kc1 Project

		LASER	CP NB	Burak Filter	WP NB
PD	*p*-value	**<<0.001**	<<0.001	<<0.001	<<0.001
	A-statistic	**1.000**	0.000	0.000	0.000
PF	*p*-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	1.000	**0.000**	**0.000**	**0.000**
Bal	*p*-value	**<<0.001**	<<0.001	<<0.001	**<<0.001**
	A-statistic	**1.000**	0.000	0.000	**0.890**

Table A3. Comparison of HISNN with Other Classification Models for the kc2 Project

		LASER	CP NB	Burak Filter	WP NB
PD	*p*-value	**<<0.001**	<<0.001	<<0.001	<<0.001
	A-statistic	**1.000**	0.000	0.000	0.000
PF	*p*-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	1.000	**0.000**	**0.000**	**0.020**
Bal	*p*-value	**<<0.001**	<<0.001	<<0.001	<<0.001
	A-statistic	**1.000**	0.000	0.000	0.030

Table A4. Comparison of HISNN with Other Classification Models for the kc3 Project

		LASER	CP NB	Burak Filter	WP NB
PD	*p*-value	**<<0.001**	<<0.001	<<0.001	<<0.001
	A-statistic	**1.000**	0.000	0.000	0.010
PF	*p*-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	1.000	**0.000**	**0.000**	**0.020**
Bal	*p*-value	**<<0.001**	**<<0.001**	**<<0.001**	0.434
	A-statistic	**1.000**	**1.000**	**1.000**	0.530

Table A5. Comparison of HISNN with Other Classification Models for the mc2 project

		LASER	CP NB	Burak Filter	WP NB
PD	p-value	**<<0.001**	<<0.001	**<<0.001**	**<<0.001**
	A-statistic	**1.000**	0.000	**1.000**	**0.660**
PF	p-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	1.000	**0.000**	**0.000**	**0.280**
Bal	p-value	**<<0.001**	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	**1.000**	**1.000**	**1.000**	**0.920**

Table A6. Comparison of HISNN with Other Classification Models for the mw1 Project

		LASER	CP NB	Burak Filter	WP NB
PD	p-value	**<<0.001**	<<0.001	0.300	**0.036**
	A-statistic	**1.000**	0.000	0.500	0.580
PF	p-value	**<<0.001**	**<<0.001**	**<<0.001**	<<0.001
	A-statistic	**0.000**	**0.000**	**0.000**	0.760
Bal	p-value	**<<0.001**	**<<0.001**	**<<0.001**	**0.036**
	A-statistic	**1.000**	**1.000**	**1.000**	0.580

Table A7. Comparison of HISNN with Other Classification Models for the pc1 Project

		LASER	CP NB	Burak Filter	WP NB
PD	p-value	**<<0.001**	<<0.001	<<0.001	<<0.001
	A-statistic	**1.000**	0.000	0.000	0.070
PF	p-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	1.000	**0.000**	**0.000**	**0.100**
Bal	p-value	<<0.001	**<<0.001**	**<<0.001**	**<<0.001**
	A-statistic	0.000	**1.000**	**1.000**	**0.880**