Information Sciences 583 (2022) 332–363

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

IVKMP: A robust data-driven heterogeneous defect model based on deep representation optimization learning

Kun Zhu [a], Shi Ying [a,*], Weiping Ding [b], Nana Zhang [a], Dandan Zhu [c]

[a] School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China
[b] School of Information Science and Technology, Nantong University, Nantong 226019, China
[c] Artificial Intelligence Institute, Shanghai Jiaotong University, Shanghai 200240, China

A R T I C L E I N F O

Article history:
Received 4 May 2021
Received in revised form 1 November 2021
Accepted 12 November 2021
Available online 20 November 2021

Keywords:
Heterogeneous defect prediction
Deep neural network
Information maximizing GANs
Principal component analysis network
multi-objective optimization

A B S T R A C T

Heterogeneous defect prediction (HDP) aims to transfer informative knowledge, namely the defect-proneness tendency of software metrics, from a source project to predict potential defects in a target project by matching metrics with similar distributions between different software projects. Nevertheless, the complex internal intrinsic structure hidden behind the defect data makes it difficult for the prior heterogeneous defect models to capture and migrate the most informative software metrics, and severely hinders HDP performance. To address these issues, we propose a robust data-driven HDP model called IVKMP in this study. We firstly adopt an advanced deep generation network – InfoGAN (Information maximizing GANs) for data augmentation, namely simultaneously achieving class balance and generating sufficient defect instances. Secondly, the multi-objective VaEA (Vector angle-based Evolutionary Algorithm) optimization is employed to select the fewest representative metric subsets while achieving the minimum error. Finally, a deep defect predictor for HDP based on the lightweight but effective deep network – PCANet (Principal Component Analysis Network) with the binary hashing and block-wise histogram is built to essentially capture more semantically related robust representations. We compare the IVKMP model with multiple state-of-the-art baseline models across 542 heterogeneous project pairs of 26 software projects. Experimental results demonstrate the superiority and robustness of our IVKMP model.

1. Introduction

Software Defect Prediction (SDP) aims to predict defect-prone software modules (instances) in advance by constructing an effective prediction model, thereby reasonably assigning priority to software quality assurance work [64]. At present, the existing SDP models are mainly divided into Within-Project Defect Prediction (WPDP) and Cross-Project Defect Prediction (CPDP) according to the actual application scenario [30]. For WPDP, an effective prediction model can be trained using the labeled modules in a software project, and then the model is used to predict whether unlabeled modules in the same project are defective [52]. However, in practical applications, due to limited data in some software projects, the WPDP cannot be widely promoted [21]. CPDP refers to constructing a prediction model for unlabeled software modules in a target project

* Corresponding author.
E-mail addresses: kkfd02@163.com (K. Zhu), yingshi@whu.edu.cn (S. Ying), dwp9988@163.com (W. Ding), nlznn92@163.com (N. Zhang), ddz@sjtu.edu.cn (D. Zhu).

https://doi.org/10.1016/j.ins.2021.11.029
0020-0255/© 2021 Published by Elsevier Inc.

using labeled historical data from other projects (i.e., source projects) [56]. Many typical CPDP approaches require that software metric sets should be identical between the source project and target project. Nevertheless, there are rarely the common metrics between different software projects in most cases, and it is not easy to find the source project that has the same metrics as the target project. For example, projects in the AEEEM dataset have 61 metrics and projects in the SOFTLAB dataset have 29 metrics, yet they have only one common metric, i.e., lines of code (LOC). Since many information-rich and representative metrics are not in the common metric set between the source project and the target project, the constructed prediction model cannot achieve satisfactory prediction performance [6]. Accordingly, Heterogeneous Defect Prediction (HDP) approaches [5,35] are proposed to predict potential defects in new software modules across projects with heterogeneous metrics (namely the defect metrics between the source project and the target project are different).

In practice, for most software projects, the number of defective modules (minority classes) is far less than the number of non-defective modules (majority classes), i.e., defect class imbalance problem. Previous studies [50,62,45,15] have shown that class imbalance can adversely affect or even seriously degrade the performance of prediction models. In addition, the source project used to predict the target project may not have enough available data for training the HDP model. Nevertheless, the lack of training data will also severely restrict the prediction performance of the HDP model. To settle these two issues, we employ a powerful deep generation model – InfoGAN (**Info**rmation maximizing **G**enerative **A**dversarial **N**ets) [4] to learn interpretable and high-level disentangled feature representations hidden in the original defect data by maximizing the mutual information between a small subset of the latent variables and the observation, so as to conduct defect class imbalance processing and generate more software defect instances for source software projects in this paper. For each source software project, we first employ InfoGAN to generate sufficient instances for non-defective and defective instances simultaneously, and then randomly select the same number of defective instances and non-defective instances, and finally replace the same number of generated defective and non-defective instances as the two types of instances in the original project with all defective and non-defective instances in the original software project respectively, thereby achieving defect class balance and providing adequate training data for the latter deep learning predictor. In this study, the deep learning generation model – InfoGAN is innovatively introduced into the field of HDP, which provides a new feasible solution for software developers or testers to simultaneously address the challenges of defect class imbalance and training data shortage.

The performance of HDP also largely depends on the quality of defect metrics in the source project. Therefore, we should select the optimal representative defect metric subsets that can be better used for the subsequent metric matching and transfer for each source software project. Since the defect metrics in different datasets may be heterogeneous distinctly and even at different granularities, we need to transfer the matched defect metrics with similar distributions from the source project to the target project. We first leverage Kolmogorov–Smirnov Test (KS-Test) [12] to measure the similarity of each metric pair between the source and target projects and calculate matching scores for all metric pairs, and then employ the maximum weighted bipartite matching [13] approach to select a set of matched metrics with the highest sum of matching scores and no duplicate metrics, thus addressing the issue of data heterogeneity in HDP. Note that some traditional metric subset selection approaches [59], such as chi-square, information gain, fisher score, can only select a metric subset for source project at a time. Unfortunately, when conducting metric matching using this approach, this only metric subset cannot match the target project in rare cases. Compared with these single-objective metric subset selection approaches, the multi-objective optimization approaches can select multiple sets of metric subsets (multiple solutions) for the source software project. When conducting metric matching, we first select the metric subset (i.e., the solution) with the first index number to match the target project. If this selected first metric subset cannot match the target project, we try to use the metric subset with the second index number to match the target project, and execute this step in turn until completing matching. Accordingly, the multi-objective metric selection approach can well compensate for the deficiency of this metric matching and transfer approach used that cannot find suitable matching objects at one time, which fully reflects the superiority of the multi-objective metric subset selection approach. This is an important reason for motivating us to conduct metric subset selection using the multi-objective optimization approach.

However, for many multi-objective metric subset selection approaches, especially for decomposition-based approaches, they have to specify a group of weight vectors that significantly affect the performance of the approach in terms of diversity, and the construction of the weight vector is quite difficult [57]. Moreover, it is challenging to maintain the uniformity for intersection points of the specified search directions and the Pareto Front (PF). In particular, for a highly irregular PF (e.g., a degenerate or discontinuous front), the uniformly-distributed weight vectors cannot ensure the uniformity of the intersection points [28,57]. To address these two limitations, we employ the multi-objective VaEA (**V**ector **A**ngle-based **E**volutionary **A**lgorithm) algorithm [57] to build a novel metric subset selection approach for source software projects in HDP. This approach performs a search for solutions (i.e., metric subsets) according to the search directions of the current evolutionary population. Since the current population is an approximation of the true PF, this approximation becomes more accurate as the population evolves [31]. Therefore, this approach does not require weight vectors. In addition, this metric subset selection approach utilizes the maximum-vector-angle-first mode in the environmental selection to ensure the uniformity and wideness of the solution set and adopts the worse-elimination mode to allow worse solutions to be conditionally substituted by other individuals, thus maintaining a good balance between the diversity and convergence of solutions (i.e., defect metric subsets). This multi-objective metric subset selection approach can gather multiple high-quality and representative metric subsets for each source project in HDP. This search based software engineering approach is promising because it can provide automated or semi-automated solutions for metric subset selection problem in HDP with large-scale complex problem spaces, which have multiple competing or even conflicting objectives. In this study, our multi-objective VaEA metric subset selection approach mainly considers two optimization objectives into the metric subset selection problem in HDP. One objective is to minimize the number of selected defect metrics, which is concerned with the cost of the constructed model. Another objective is to minimize the error of this model, which involves in the benefit of the model. Whereas, since the VaEA algorithm is regarded as a wrapper based metric subset selection approach, we should select a suitable classifier to evaluate the performance of the approach and embed the classifier into the objective function. Simultaneously, considering the computational complexity and feasibility of the objective function, it is impractical to select a complex classifier, thus we choose the classic k-nearest neighbor (KNN) algorithm as the classifier of the selection approach.

Previous studies [37,54] have demonstrated that an excellent metric representation plays a critical role in the prediction performance of a model. Motivated by these studies, we aim to find a strong distinguishing feature representation for each software project in HDP, which not only requires the metrics having the strong discriminant inter-class separability, but also reserves unaltered intra-class compactness [17]. In recent years, due to the capability to extract abstract semantic metrics with such strong discriminative capability compared to traditional machine learning approaches, a lightweight but effective deep learning network – PCANet (Principal Component Analysis Network) with SVM (Support Vector Machine) [1], has gained significant attention and been successfully applied in image classification. For defect metrics in HDP, the PCANet with the binary hashing and block-wise histograms can learn sufficient robustness and invariance, which is capable of essentially capturing more semantically related abstract representations. Prior studies [49,16] have demonstrated that the abstract deep semantic features have stronger discriminating capacity for different classes (defective or non-defective). Furthermore, such a lightweight deep neural network is well suited to our relatively small software defect projects. Consequently, we employ this lightweight but effective deep learning network – Principal Component Analysis Network (PCANet) with SVM [1] to construct a prominent defect predictor in HDP, in which the PCANet contains cascaded principal component analysis (PCA), binary hashing and blockwise histograms. Chan et al. [1] have demonstrated that the PCANet can achieve very competitive results in many classification tasks, such as texture classification, object recognition, which are already on par with, or often better than, state-of-the-art approaches. This PCANet predictor consists of three processing stages. In the first two stages, the basic PCA filters are utilized to learn the two-stage convolution filter banks instead of the convolution operation in the traditional convolutional neural network. The first two stages are linear operations without using complex regularization parameters and optimization procedures. In the last output layer, we adopt the binary hashing for encoding and nonlinear processing, and use block-wise histograms of the binary hashing encoding for pooling operations. The output of the block-wise histograms is regarded as the final output metric of the PCANet. Finally, the SVM classifier is employed to predict whether each instance module is defective by discriminating the robust semantic features extracted.

Based on the above analysis, we propose a novel data-driven heterogeneous defect prediction model called IVKMP (a pipeline model concatenated by **I**nfoGAN, multi-objective **V**aEA optimization, **K**STest, **M**aximum weighted bipartite matching and **P**CANet) in this study. The IVKMP model driven by defect metric stream is able to find the matching metrics with the same distribution from the heterogeneous metrics between the source project and the target project. The model employs a powerful deep generation model – InfoGAN to conduct data augmentation tasks, which provides a new feasible solution for software developers or testers to simultaneously effectively address the challenges of defect class imbalance and training data shortage in realistic source software projects. In addition, the multi-objective VaEA optimization approach is adopted to select the fewest representative balanced solutions (i.e., defect metric subsets) between the diversity and convergence for the minimum error by the maximum-vector-angle-first mode and the worse-elimination mode, so as to more precisely eliminate irrelevant and redundant defect metrics for source software projects and well compensate for the deficiency of this metric matching and transfer approach used that cannot find suitable matching objects at one time. Furthermore, a deep defect predictor for HDP based on the lightweight but effective deep learning network – PCANet with the binaryhashing and block-wise histogram is built to learn more robust semantic metric representations with strong discriminating capacity. In consequence, our IVKMP model can well address the issue of the heterogeneous metric sets with different distributions, which can benefit developers or testers who want to build a high-precision heterogeneous prediction model with more defects from publicly available open source software defect datasets even whose source code is not available. This HDP model can broaden the channels of training data required by the SDP model and boost the probability that a defect hidden in the software project is detected. As an important software quality assurance means, the IVKMP model can more effectively help developers or testers to save testing cost by predicting the potentially defective modules in advance and allocate limited software maintenance resources reasonably, and also improves the quality of software products.

In this paper, the main contributions can be summarized as follows:

(1) We propose a robust data-driven HDP model called IVKMP based on deep learning techniques. The model adopts an advanced deep generation network – InfoGAN for data augmentation, namely simultaneously achieving defect class balance and generating sufficient defect instances for source software project. In addition, we build a deep defect predictor for HDP based on the lightweight but effective deep learning network – PCANet with the binary hashing and block-wise histograms in the IVKMP model, which can essentially capture more semantically related robust metric representations.

(2) The model also employs the multi-objective VaEA optimization to select the fewest representative metric subsets while achieving the minimum error for source projects. To our best knowledge, this is the first work to investigate the metric subset selection performance of multi-objective optimization algorithms in HDP.

(3) We conduct extensive and large-scale experiments to evaluate the HDP performance of the IVKMP model compared with multiple state-of-the-art baseline models across 542 heterogeneous project pairs of 26 software projects. We also compare the VaEA optimization approach in IVKMP with eight state-of-the-art multi-objective metric subset selection approaches,

and compare the PCANet predictor in IVKMP with seven classic defect predictors on HDP performance. Experimental results verify that our model can achieve very competitive HDP performance.

The reminder of this paper is organized as follows: Section 2 introduces the related work. Section 3 details the proposed IVKMP model. Section 4 shows the experimental setup, including experimental subjects, evaluation indicators, experimental design, parameter settings, and statistical significance test. Section 5 evaluates the HDP performance of our IVKMP model. Section 6 discusses the possible reasons that affect the HDP performance of our IVKMP model. We conclude the paper and present the future work in Section 7.

2. Related work

In this section, we describe the typical heterogeneous defect prediction models and deep learning models in software engineering.

2.1. Heterogeneous defect prediction

In recent years, heterogeneous information processing techniques [41,40] have gained significant attention and been successfully applied in many fields, such as information recommendation [39], micro-video understanding [53]. Shi et al. [39] put forward a heterogeneous network embedding approach guided by meta-paths to reveal the structural and semantic information of heterogeneous information networks. Moreover, they also devised a heterogeneous information network embedding for recommendation model named HERec, which can effectively incorporate different heterogeneous embedding information to boost the recommendation performance. Wei et al. [53] proposed a deep multimodal cooperative learning method that can be applied to the venue category estimation of micro-videos, which can automatically discriminate and fuse the complementary and the consistent information among multiple modalities.

Similarly, in the field of HDP, a few classic prediction models [61,35,20,26] have been proposed. These HDP models are mainly divided into two types: metric selection and matching based HDP models; metric transformation based HDP models.

For metric selection and matching based HDP models, they mainly adopt some traditional metric selection approaches, such as correlation-based Feature Selection (CFS) approach with best-first search [61], chi-square, gain ratio, relief-F [35], and some classic metric matching approaches, including distance of feature distribution curves [61], PAnalyzer, KSAnalyzer, SCoAnalyzer [35]. Yu et al. [61] proposed a novel Feature Matching and Transfer (FMT) method. This method uses the correlation-based Feature Selection (CFS) approach with best-first search [19] to perform metric selection for the source project, and obtains the distribution curves of the selected metrics in the source project and all the metrics in the target project, and then presents a feature matching algorithm to transform the heterogeneous metrics into the matched metrics. The experimental results showed that the FMT method is quite effective. Nam et al. [35] proposed a new HDP method named HDP-KS, which first uses the metric selection approaches (such as chi-square, gain ratio, relief-F, significance attribute evaluation) to eliminate redundant and irrelevant metrics for source projects, and then utilizes three metric matching approaches, i.e., Percentile (PAnalyzer), Kolmogorov–Smirnov Test (KSAnalyzer) and Spearman's Correlation (SCoAnalyzer) to match up metric differences between the source and target projects. Experimental results showed that the KSAnalyzer (i.e., HDP-KS) approach can achieve the best prediction performance.

For metric transformation based HDP models, these models mainly adopt some typical metric transfer approaches, such as distribution characteristics [20], manifold discriminant alignment (MDA) [30], canonical correlation analysis (CCA) [6,26]. He et al. [20] proposed a HDP model called CPDP-IFS based on 16 distribution characteristics of metric values of each defect instance, such as standard deviation, interquartile range, skewness, variance. Li et al. [29] presented a new Two-Stage Ensemble Learning (TSEL) HDP approach, and this approach consists of two stages: Ensemble Multi-kernel Domain Adaptation (EMDA) stage and Ensemble Data Sampling (EDS) stage. The experimental results proved that this TSEL approach can achieve 20.14%-33.92%, 36.05%-54.78%, 5.48%–19.93% improvements in terms of AUC, F-measure, balance, respectively. Li et al. [30] also designed a multi-source heterogeneous prediction model named Multi-source Selection approach based Manifold Discriminant Alignment (MSMDA), which can select multiple source projects with similar distributions for target projects, and proposed a privacy preservation method called Sparse Representation based Double Obfuscation (SRDO). In addition, Cheng et al. [6] proposed a Canonical Correlation Analysis (CCA) approach used for calculating a joint feature space of cross-project data and presented a new support vector machine approach that can merge the correlation transfer knowledges and different misclassification costs into the built classifier. Jing et al. [26] presented a Unified Metric Representation (UMR) that measures heterogeneous data between the source company and the target company, and proposed a new transfer learning approach called CCA + based on canonical correlation analysis that can make the data distribution more similar between the source company and the target company.

Different from these traditional metric selection approaches, we employ the multi-objective VaEA optimization approach to select the fewest representative metric subsets while achieving the minimum error for source projects, which not only has the strong metric selection capacity, but also well compensates for the deficiency of this metric matching and transfer approach (i.e., KSTest, maximum weighted bipartite matching) used that cannot find suitable matching objects at one time. In addition, these HDP models mainly use some traditional machine learning methods as defect predictors, such as LR (Logistic Regression), NB (Naïve Bayes), SVM (Support Vector Machine). Compared with these traditional machine learning methods, we build a deep defect predictor based on the lightweight but effective deep network – PCANet with the binary hashing and block-wise histograms to learn more robust semantic metric representations, and these representations have stronger discriminating capacity for different classes (defective or non-defective). We also adopt an advanced deep generation network – InfoGAN for data augmentation, which can simultaneously achieve defect class balance and generate sufficient defect instances for source software project.

2.2. Deep learning models in software engineering

In recent years, researchers have adopted deep learning techniques (e.g., CNN, DBN) for addressing some typical issues in software engineering. Zhou et al. [68] used the deep forest technique to build an effective defect predictor, which can detect the most important features through a cascaded layer-by-layer forest structure. The experimental results showed that the cascaded deep forest can boost the AUC indicator by 7% on average compared to the deep forest without the cascade strategy and 8% on average compared to the other deep learning model – Deep Belief Network (DBN). Wang et al. [49] utilized Deep Belief Network (DBN) to extract token vectors from the program's Abstract Syntax Tree (AST), thus automatically learning abstract semantic features. The experimental results verified that these semantic features significantly enhance the prediction performance of Within-Project Defect Prediction(WPDP) and Cross-Project Defect Prediction (CPDP) on ten publicly available software projects. Zhu et al. [71] presented a novel WPDP model called DLDD based on hybrid deep learning techniques, which utilizes denoising autoencoder to learn robust features that are not contaminated by noise and adopts Deep Neural Network (DNN) for learning deep semantic features. Ferrari et al. [11] proved that which extent Natural Language Processing (NLP) could be practically used for detecting potential defects in the requirement documents of the railway signalling manufacturer. Xu et al. [58] proposed a novel defect detection network (D4Net), which focuses on differences between high-level semantic features extracted from the deep neural network to capture the region of possible defects. Experimental results verified that the network can achieve the optimal performances of 96.9% accuracy and 91.7% F-measure in a real industrial dataset. Huo et al. [24] utilized Convolutional Neural Network (CNN) to learn the metric representation from the source code and the text in the defect report, and then combined two kinds of metrics as a kind of unified metric representation for defect location. Ha and Zhang [18] leveraged deep Feedforward Neural Network (FNN) and the L1 regularization to predict the performance for highly configurable software systems. The experimental results showed the superiority of this approach on eleven open source datasets. Ghaffarian and Shahriari [14] presented a neural vulnerability analysis method, which can employ customized intermediate graph representations of programs to train graph neural network. Experimental results on a public suite of vulnerable programs proved that their method can achieve very competitive performance. In addition, some typical deep learning techniques have also been used for mutation testing [32], code clone detection [51], comment completions [7], security-related issues identifying [36] and so on.

As discussed, although much effort has been dedicated to metric representations and performance enhancement for deep learning models in software engineering, the noted models suffer from the following limitations and challenges: (1) These algorithms do not investigate the issues of class imbalance and data shortage, which are still common problems in software engineering. (2) Irrelevant or redundant metrics are still not well removed, which severely hinders model performance. (3) The generalization capability and robustness of feature representations extracted by these algorithms still need to be further enhanced.

3. The proposed IVKMP model

In this study, we build a robust data-driven heterogeneous defect prediction model called IVKMP, which is a concatenated pipeline model driven by defect metric stream. The model is able to find the matching metrics with the same distribution from the heterogeneous metrics between the source project and the target project. The IVKMP model is composed of four steps: (1) Defect data augmentation based on InfoGAN; (2) Multi-objective VaEA optimization based metric subset selection; (3) Metric matching and transfer; (4) Heterogeneous defect prediction based on PCANet. Next we will introduce the proposed IVKMP model in detail. The flowchart of our IVKMP model is shown in Fig. 1.

3.1. Defect data augmentation based on InfoGAN

In HDP, most software projects usually present class imbalance phenomenon. In other words, the number of defective modules (minority classes) is far less than the number of non-defective modules (majority classes). Furthermore, the source software projects used to predict the target project may not have sufficient training data. Both of the above issues severely limit the prediction performance of the HDP model. Therefore, as the first step of our IVKMP model, we employ InfoGAN [4] to perform data augmentation for each of the original software projects separately, namely generating more software defect instances and achieving defect class balance in this paper. For each source software project, we first utilize InfoGAN to generate sufficient instances for non-defective and defective instances simultaneously, and then randomly select the same number of defective instances and non-defective instances, and finally replace the same number of generated defective and non-defective instances as the two types of instances in the original project with all defective and non-defective instances in the original software project respectively, thereby achieving defect class balance and providing adequate training data for the

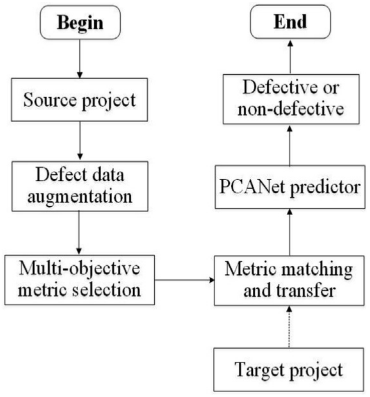

Begin
Source project
Defect data augmentation
Multi-objective metric selection
Metric matching and transfer
Target project
PCANet predictor
Defective or non-defective
End

Fig. 1. The flowchart of the proposed IVKMP model.

latter deep learning predictor. In actual software development, this approach can provide a new feasible solution for software developers or testers to simultaneously address the challenges of defect class imbalance and training data shortage.

InfoGAN can learn interpretable and meaningful disentangled feature representations hidden in the original defect data by maximizing the mutual information between a small subset of the latent variables and the observation. The training goal of InfoGAN is to learn a generator distribution $P_G(x)$ that can match the real data distribution $P_{\text{data}}(x)$ as much as possible, and we just utilize this idea to conduct defect instance generation and class imbalance processing in this paper. Considering that directly maximizing the mutual information $I(\vartheta|X)$ needs additional storage and computing cost, InfoGAN adopts the variational mutual information maximization to compute lower bounding mutual information by defining an auxiliary distribution (i.e., recognition network) $S(\vartheta|X)$ to approximate the posterior $P(\vartheta|X)$, in which X denotes the generated data and ϑ denotes the latent code. Furthermore, the InfoGAN can share the neural network between recognition network $S(\vartheta|X)$ and the discriminator $D(x)$, and can maximize the mutual information $I(\vartheta|X = G(\sigma,\vartheta))$ between the generated data X and the latent code ϑ, where σ represents the noise variable. To further get rid of sampling from the posterior $P(\vartheta|X)$, InfoGAN utilizes the Lemma theorem [4] to derive the variational lower bound $J_I(G,S)$ of the mutual information $I(\vartheta|X = G(\sigma,\vartheta))$, as shown in Eq. (1):

$$J_I(G,S) = \mathbb{E}_{X\sim P_{G(\sigma,\vartheta)}(X),\vartheta\sim P(\vartheta|X=x)}[\ln S(\vartheta,x)] + L(\vartheta) \leqslant I(\vartheta|X = G(\sigma,\vartheta)), \tag{1}$$

where $J_I(G,S)$ can be approximated with Monte Carlo simulation, and $L(\vartheta)$ denotes the entropy of latent code ϑ.

Accordingly, InfoGAN adds the variational regularization term $J_I(G,S)$ of mutual information $I(\vartheta|X = G(\sigma,\vartheta))$ to the objective function of InfoGAN with a hyperparameter θ:

$$\min_{G,S}\max_{D} V_{\text{InfoGAN}}(G,D,S) = V(G,D) - \theta J_I(G,S). \tag{2}$$

Through the above learning process, we can employ the InfoGAN to generate more defect instances and fulfill class imbalance processing, thereby acquiring the generated defect data D for each source software project and providing sufficient training data for the later deep learning model.

Similar to the original software defect data distributions, the defect metric distributions in different dimensions generated by InfoGAN are also of large difference. Hence, we further leverage the z-score approach [60] to perform data standardization for these defect metrics, and obtain new defect dataset D' for each source software project.

3.2. Multi-objective VaEA optimization based metric subset selection

We mainly focus on metric subset selection for source software projects used for training in this paper, which aims to identify and eliminate irrelevant or redundant defect metrics as many as possible by selecting a representative metric subset that can reflect the complex intrinsic structure hidden behind the original defect metrics. Motivated by the idea of search based software engineering that can provide automated or semi-automated solutions for many practical optimization problems with large-scale complex problem space, we utilize an advanced multi-objective intelligent optimization algorithm – VaEA for metric subset selection, so as to formalize metric subset selection for HDP as a multi-objective optimization problem.

In this section, we first formalize the multi-objective optimization problem for metric subset selection in HDP. Next, we provide metric subset selection process based on multi-objective VaEA algorithm in detail.

3.2.1. The multi-objective metric subset selection optimization problem

Most source software projects for HDP contain a large number of defect metrics, among which there exist some irrelevant or redundant metrics that are not conducive to the built prediction model. To enhance the HDP performance and decrease the complexity of the prediction model, it is desirable to select only closely relevant and representative metrics that can reflect the inherent structure information hidden behind the defect data. Consequently, we employ a multi-objective optimization algorithm – VaEA to conduct metric subset selection for source software projects, which mainly considers two optimization objectives into the metric selection problem. One objective is to minimize the number of selected metrics as much as possible, which is related to the cost of the model. Another objective aims to minimize the error of the model, which is associated with the benefit of the model.

In the metric selection approach, wrapper-based approachs usually iteratively select metrics for achieving the optimal performance by a specific classification or regression algorithm [33]. Since our VaEA algorithm is regarded as a wrapper based metric subset selection approach, we should select a suitable classifier to evaluate the performance of the approach and embed the classifier into the objective function of the approach. Simultaneously, considering the computational complexity and feasibility of the objective function, it is impractical to select a complex classifier, thus we adopt the classic k-nearest neighbor (KNN) algorithm as the classifier of the multi-objective VaEA algorithm, in which the parameter k is set to 3 in this paper, and the specific tuning process will be discussed in Section 5.1.3.

The metric subset selection problem can be transformed into a bi-objective MOP by defining the following two objective functions:

$$\mathrm{Min}\,\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))^{\mathrm{T}}, \tag{3}$$

$$f_1(\mathbf{x}) = E(\mathbf{x}) = \tfrac{1}{n}\sum_{i=1}^{n}\mathrm{II}(f(x_{\mathrm{i}}) \neq y_{\mathrm{i}}), \tag{4}$$

$$f_2(\mathbf{x}) = |\mathbf{x}_l|, \tag{5}$$

where $\mathbf{x}$ denotes the defect metrics in the defect data D' processed by InfoGAN and z-score. $E(\mathbf{x})$ represents the error of the multi-objective subset selection optimization approach. $\mathbf{x}_l$ denotes the length of the selected defect metrics. f_1 aims to minimize the error and f_2 aims to minimize the number of metrics.

We adopt the binary encoding (i.e., 0 or 1) for individual encoding, in which each decision variable indicates whether a metric is selected (The value 1 represents that the ith metric can be selected, otherwise not.). The length of the decision vector is equal to the total number of defect metrics in each software project.

3.2.2. Metric subset selection approach based on multi-objective VaEA optimization

The multi-objective VaEA metric subset selection approach considers convergence and diversity simultaneously and expects to strike a good balance between them, which can utilize the maximum-vector-angle-first mechanism to select solutions from the critical layer one by one in terms of diversity, and employ the worse-elimination mechanism to allow worse solutions to be conditionally substituted by other individuals in terms of convergence. In addition, this approach has the following three merits: no need to preset weight vectors, lower algorithm complexity, fewer computational parameters. In the multi-objective VaEA metric subset selection approach, except for some common genetic algorithm parameters, such as the population size, termination criterion and parameters in genetic operators, and some common parameters used to metric selection, such as the number of objectives and the number of decision variables and the parameter k in k-nearest neighbor (KNN), there is no need to set any new special parameters. Compared with our multi-objective VaEA metric subset selection approach, many other approaches need to set new special parameters. For example, the MOEA-D-DU [63] approach also needs to set two new special parameters, namely the probability of choosing parents locally and the number of nearest weight vectors. For KnEA [67], the rate of knee points in the population needs to be set. The NSLS approach also needs to set two new special parameters, namely the mean value of the Gaussian distribution and the standard deviation of the Gaussian distribution. For RM-MEDA [66], the number of clusters needs to be set.

The pseudo code of the multi-objective VaEA optimization for metric subset selection is shown in Algorithm 1. We first randomly initialize a population with N solutions in the entire decision space (line 1). Next we calculate two objective values for each individual, i.e., the number of metrics and the classification error on the training set (line 4). In line 5, more potential feasible solutions are selected into the mating pool in accordance with the fitness value of each individual. We employ the crossover and mutation operations to generate a group of offspring solutions R, and further form a new solution U by merging P and R (lines 6–8). In lines 9–31, we select N solutions from the union set U through an environmental selection mechanism. We normalize the union set U to form the normalized objective space U' in line 10, as shown in Eqs. (6), (7):

$$f'_{\mathrm{i}}(\mathbf{x}_{\mathrm{k}}) = \frac{f_{\mathrm{i}}(\mathbf{x}_{\mathrm{k}}) - h_{\mathrm{i}}^{\min}}{h_{\mathrm{i}}^{\max} - h_{\mathrm{i}}^{\min}}, i = 1, 2, \ldots, m, \tag{6}$$

$$\mathbf{F}'(\mathbf{x}_k) = \left(f'_1(\mathbf{x}_k), f'_2(\mathbf{x}_k), \ldots, f'_m(\mathbf{x}_k)\right)^T, \tag{7}$$

Algorithm 1: Pseudo-code of the multi-objective VaEA optimization for metric subset selection

```
Input: population size: $N$, initial population: $P_0$, maximum generation: $M_{max}$
Output: The optimal metric subsets: $D''$ and training and testing classification error for each source project
1: $P_0 = InitPop(P)$;
2: $M = 1$;
3: for $M \leftarrow 1$ to $M_{max}$ do
4:    Calculate two objective values for each individual;
5:    $\overline{P} = SelectMating(P)$;
6:    $Q = Crossover(\overline{P})$;
7:    $R = Mutation(Q)$;
8:    $U = P \cup R$;
9:    P=Ø, i = 1;
10:    $U'=Normalization(U)$;
11:    $U''=CalculateNorm(U')$;
12:    $(F_1, F_2, \ldots, F_t)$ = Non-dominated-sort$(U'')$;
13:    for $|P| + |F_i| \leftarrow 1$ to $N$ do
14:       $P = P \cup F_i$;
15:       $i \leftarrow i + 1$;
16:    end for
17:    Set the critical layer: $F_t = F_i$;
18:    if $|P| = N$ then
19:       return $P$;
20:    else
21:       $L = N - |P|$;
22:       $Associate(P, F_t)$;
23:       $I = |F_t|$;
24:       $flag(\mathbf{x}_k) = flase, k \in [1, I]$;
25:       for $l \leftarrow 1$ to $L$ do
26:          $\delta = \arg\max\{\varphi(\mathbf{x}_k) \mid k \in [1, I]\}$;
27:          $\eta = \arg\min\{\varphi(\mathbf{x}_k) \mid k \in [1, I]\}$;
28:          $P$ = Maximum-Vector-Angle-First $(P, \delta, I)$;
29:          $P$ = Worse-Elimination $(P, \eta, I)$;
30:       end for
31:    end if
32: end for
33: Compute the classification error of the non-dominated solutions (metric subsets) $D''$ on the test set;
34: return The optimal solutions (metric subsets) $D''$ and training and testing classification error.
```

where the nadir point $\mathbf{H}^{max} = \left(h_1^{max}, h_2^{max}, \ldots, h_m^{max}\right)^T$ and the ideal point $\mathbf{H}^{min} = \left(h_1^{min}, h_2^{min}, \ldots, h_m^{min}\right)^T$ can be calculated by searching the maximum value and the minimum value of each objective for all solutions in the union set U, respectively, namely $h_i^{max} = \max_{k=1}^{2N} f_i(\mathbf{x}_k)$ and $h_i^{min} = \min_{k=1}^{2N} f_i(\mathbf{x}_k)$. The objective vector $\mathbf{F}(\mathbf{x}_k)$ of the solution $\mathbf{x}_k$ is normalized to form $\mathbf{F}'(\mathbf{x}_k)$.

Meanwhile, the fitness value fit$(\mathbf{x}_k)$ that can distinguish which one to retain between two closest solutions according to the vector angle in the objective space is calculated as follows:

$$\text{fit}(\mathbf{x}_k) = \sum_{i=1}^{m} f'_i(\mathbf{x}_k). \tag{8}$$

In line 11, we calculate the norm of each solution $\mathbf{x}_k$ in the normalized objective space U', as shown in Eq. (9):

$$\text{norm}(\mathbf{x}_k) = \sqrt{\sum_{i=1}^{m} f'_i(\mathbf{x}_k)^2}. \tag{9}$$

The vector angle between two solutions $\mathbf{x}_k$ and $\mathbf{z}_t$ is calculated as follows:

$$A(\mathbf{x}_k, \mathbf{z}_t) = \arccos \left| \frac{\sum_{i=1}^{m} f'_i(\mathbf{x}_k).f'_i(\mathbf{z}_t)}{\text{norm}(\mathbf{x}_k).\text{norm}(\mathbf{z}_t)} \right|. \tag{10}$$

We can divide the solutions into different layers through non-dominated sorting and obtain the critical layer F_t (lines 12–17). If $|P| = N$, then P is returned (lines 18–19). Otherwise, $L = N - P$ solutions from the critical layer F_t are stacked into P one by one by association and niche preservation operations according to vector angles (lines 20–30). In lines 23–30, the niche preservation operation takes the diversity and convergence of the population into consideration simultaneously according to the maximum-vector-angle-first and worse-elimination mechanism, respectively. Each solution in the critical layer F_t has a flag value that can denote whether a solution is stacked into the population P, which is initialized as *false* (lines 23–24). The index of each solution in the critical layer F_t has the maximum and minimum vector angles denoted by δ and η, respectively (lines 26–27). In line 28, the goal of the maximum-vector-angle-first mechanism is to find the optimal solution with the maximum vector angle in terms of diversity and add this solution into the population P. If δ is null, then P is returned (all solutions have been added to P). Otherwise, $\mathbf{x}_\delta$ is stacked into the population P and its flag is transformed into to *true*, and the vector angles of the remaining solutions in critical layer F_t should be updated. In line 29, the worse-elimination mechanism can allow worse solutions to be conditionally substituted by other solutions in terms of convergence, thus achieving a balance between diversity and convergence. If the angle between $\mathbf{x}_\eta$ and its target solution $\mathbf{y}_q$ is smaller than $\gamma = ((\pi/2)/N + 1)$, meanwhile, when $\mathbf{x}_\eta$ is not added to the population P and the fitness value of $\mathbf{x}_\eta$ is better than target solution $\mathbf{y}_q$, and $\mathbf{y}_q$ can only be substituted by $\mathbf{x}_\eta$ in this case. This is because there is no need to keep multiple solutions in the same search direction. After completing substituting, we consider the following two situations: (1) for the corresponding index $\tau(\mathbf{x}_k)$ of the target solution, if $\tau(\mathbf{x}_k) \neq \tau(\mathbf{x}_\eta), \mathbf{x}_k$ and $\mathbf{x}_\eta$ are associated with different individuals in the population P, then we only need to compare whether the angle between $\mathbf{x}_k$ and $\mathbf{x}_\eta$ is smaller than $\varphi(\mathbf{x}_k)$. (2) if $\tau(\mathbf{x}_k) = \tau(\mathbf{x}_\eta), \mathbf{x}_k$ and $\mathbf{x}_\eta$ are associated with the identical individual, then we only need to update $\varphi(\mathbf{x}_k)$.

After many iterations, the VaEA algorithm will reach to the maximum generation $M_{\max}$ and converge to stable solutions. In line 33, we compute the classification error of the non-dominated solutions (metric subsets) D'' on the test set. Finally, the multi-objective VaEA optimization algorithm can return the optimal solutions (metric subsets) D'' and their training and testing classification error, thus acquiring the optimal metric subsets D'' for source software project.

Since the calculation process of the time complexity for VaEA is very complicated, we directly report the time complexity of its one generation as $O\left(mN^2\right)$, where m represents the number of objectives (m is equal to 2 in this paper) and N represents the population size. We also analyze and report the time complexity of three metric subset selection approaches inferior only to VaEA in terms of running time (see Section 5.1.2), including KnEA ($O\left(mN^2\right)$), hpaEA ($O\left(Nlog^{m-1}N + mN^2 + m^3N\right)$) and MaOEA-R&D ($O\left(m^2N + mN^2\right)$). Therefore, we can observe that the time complexity of VaEA is the same as that of KnEA, but lower than those of hpaEA and MaOEA-R&D.

3.3. Metric matching and transfer

After completing metric selection, we eliminate irrelevant or redundant defect metrics for source software projects. Nevertheless, the metric distribution between the source and target projects is different in HDP. Consequently, we need to transfer knowledge with rich information, i.e., the typical defect-proneness tendency of software metrics, from a source project to conduct defect prediction for a target project by matching metrics with similar distribution between them.

In this paper, we utilize Kolmogorov–Smirnov Test (KS-Test) [12] to assess the similarity of each metric pair between the source and target projects and compute matching scores for all metric pairs, Nam et al. [35] have verified the validity of the KS-test approach in HDP. The KS-test is a non-parametric statistical test approach for measuring the similarity between source metrics and target metrics, and regards the p-value that can reflect the significance level as the matching score, i.e., $S_{ij} = p_{ij}$, where p_{ij} denotes a p-value between ith source and jth target metrics acquired by KS-Test, and the p-value represents the significance level that the null hypothesis can be rejected by very strong evidence, i.e., two instances come from the same distribution. We select a p-value cutoff threshold of 0.05 which is the generally accepted significance level in the common statistical test problems [8]. Then we utilize the maximum weighted bipartite matching [35,13] approach to pick a set of matched metrics with the highest sum of matching scores and no duplicate metrics, and employ the picked matched metrics to construct the next prediction model.

For each source software project, the multi-objective VaEA metric subset selection approach can generate N solutions, in which each solution has an index number (i.e., 1,2,...,N) and contains two minimization optimization objectives (i.e., the minimum number of metrics and the minimum error). At the same time, corresponding to the minimum number of metrics, the selected metric subset is also determined. Although this adopted metric matching and transfer approach has strict requirements on the matching objects in the target project, it has a relatively high matching accuracy [35]. When conducting metric matching using KS-Test and the maximum weighted bipartite matching, we first select the metric subset (i.e., the solution) with index number “1” to match the target project. If this selected first metric subset cannot match the target project, we can try to use the metric subset with index number “2” to match the target project, and execute this step in turn until completing matching. Consequently, the multi-objective VaEA metric selection approach can well compensate for the deficiency of this metric matching and transfer approach used that cannot find suitable matching objects at one time, which fully reflects the superiority of the multi-objective metric subset selection approach.

Finally, the IVKMP model driven by defect metric stream is able to find the matching metrics with the same distribution from the heterogeneous metrics between the source project and the target project. We can transfer the matched defect metrics D^+ from the source project to the target project, thereby addressing the issue of metric heterogeneity in HDP.

3.4. Heterogeneous defect prediction based on PCANet

In this section, we build a deep defect predictor for heterogeneous defect prediction based on the lightweight but effective deep learning network – PCANet with the binary hashing and block-wise histograms [1] after metric matching and transfer, in which the PCANet is composed of three basic components: (1) cascaded principal component analysis (PCA); (2) binary hashing; (3) blockwise histograms. This PCANet consists of three processing stages. In the first two stages, the basic PCA filters are adopted to learn the two-stage convolution filter banks instead of the convolution operation in the traditional convolutional neural network. The first two stages are linear operations without using regularization parameters and complex optimization procedures. In the last output layer, the binary hashing approach is used for encoding and nonlinear processing, and the block-wise histograms of the binary hashing encoding are adopted for pooling operations. The output of the block-wise histograms is regarded as the output metric representation of the PCANet. The pseudo-code of the deep defect predictor PCANet is as shown in Algorithm 2.

For each source software project, given the training instances $D^{++} = \{(Z_\text{i}, Y_\text{i})|i = 1, 2, \dots, N'\}, Z_\text{i} \in \mathbb{R}^{p\times q}$ represents the ith training instance, where p and q denote the width and height of the matrix reshaped for each training instance (lines 1–2). Note that before reshaping, assume that a certain software project has a-dimensional metrics, if $a \neq b^2$, b = 1, 2, 3..., we need to fill in 0 in the last $(b^2 - a)$ columns. $Y_i \in \mathbb{R}^m$ denotes the class label (defective or non-defective) corresponding to Z_i. The filter size is $c_1 \times c_2$. We derive the three-stage process of the PCANet as follows:

In the first stage (PCA), we utilize the PCA filters to learn robust features from the training instances. On each defect instance, we employ the $c_1 \times c_2$ filter to extract all defect metrics, and a defect instance can be expressed as follows: $z_{\text{i},1}, z_{\text{i},2}, \dots, z_{\text{i},\text{p}'\text{q}'} \in \mathbb{R}^{c_1c_2}$, where $p' = p - \lceil c_1/2 \rceil, q' = q - \lceil c_2/2 \rceil$ (line 3). After removing mean (for better capturing major variations in the defect metrics) for each defect instance and concatenating all of the filter outputs, we can achieve the training instance matrix $\mathbf{Z}$ for all defect instances (lines 4–5).

The PCA algorithm aims to minimize the reconstruction error by finding a series of orthonormal matrices (filters) (line 6), as shown in Eq. (11):

$$L = \min \left\| \mathbf{Z} - \mathbf{O}\mathbf{O}^\mathbf{T}\mathbf{Z} \right\|_F^2, \tag{11}$$

where $\mathbf{O} \in \mathbb{R}^{c_1c_2\times T_1}, T_1$ denotes the number of filters in the first stage.

Algorithm 2: Pseudo-code of the PCANet defect predictor

```
Input:
  the matched defect metrics: $D^+$, the number of PCA stages: 2, the filter size: $c_1, c_2$, the number of filters in each stage:
  $T_1, T_2$, the block size for local histograms in the output layer.
Output:
  The final HDP result for each heterogeneous project pair: $R_\text{H}$.
1: Fill in 0 in the last $(b^2 - a)$ columns for $a$-dimensional metrics when $a \neq b^2$, $b$ = 1, 2, 3...;
2: Reshape each defect instance to a matrix: $D^{++} = \{(Z_\text{i}, Y_\text{i})|i = 1, 2, \dots, N'\}, Z_\text{i} \in \mathbb{R}^{p\times q}$;
3: Employ the $c_1 \times c_2$ filter to extract all defect metrics: $z_{\text{i},1}, z_{\text{i},2}, \dots, z_{\text{i},\text{p}'\text{q}'} \in \mathbb{R}^{c_1c_2}, p' = p - \lceil c_1/2 \rceil, q' = q - \lceil c_2/2 \rceil$;/* The
  first stage (PCA) (lines 3–8) */
4: Remove mean for better capturing major variations;
5: Concatenate all of the filter outputs as matrix $\mathbf{Z}$;
6: Minimize the reconstruction via a series of orthonormal filters: $L = \min \left\| \mathbf{Z} - \mathbf{O}\mathbf{O}^\mathbf{T}\mathbf{Z} \right\|_F^2, \mathbf{O} \in \mathbb{R}^{c_1c_2\times T_1}$;
7: Calculate the corresponding PCA filter: $\mathbf{J}_t^1 = \text{M}_{c_1,c_2}\left(\mathbf{u_t}\left(\mathbf{Z}\mathbf{Z}^\mathbf{T}\right)\right), t = 1, 2, \dots, T_1$ ;
8: Calculate the mapping output of the $t$th filter in the first stage: $Z_\text{i}^\text{t} = Z_\text{i} \ast \text{J}_\text{t}^1, i = 1, 2, \dots, N'$;
9: Employ the $c_1 \times c_2$ filter to extract all defect metrics;/* The second stage (PCA) (lines 9–13) */
10: Remove mean for better capturing major variations;
11: Concatenate all of the filter outputs as matrix $\mathbf{H}$;
12: Calculate the PCA filters in the second stage: $\mathbf{J}_t^2 = \text{M}_{c_1,c_2}\left(\mathbf{u_t}\left(\mathbf{H}\mathbf{H}^\mathbf{T}\right)\right), t = 1, 2, \dots, T_2$;
13: Calculate the mapping output of the $t$th filter in the second stage: $\chi_\text{i}^\text{t} = \left\{Z_\text{i}^\text{t} \ast \mathbf{J}_t^2\right\}_{\text{t}=1}^{\text{T}_2}$;
14: Binarize the $T_2$ instances $\left\{Z_\text{i}^\text{t} \ast \mathbf{J}_t^2\right\}_{\text{t}=1}^{\text{T}_2}$ in the second stage: $\left\{H\left(Z_\text{i}^\text{t} \ast \mathbf{J}_t^2\right)\right\}_{\text{t}=1}^{\text{T}_2}$;/* The output stage (binary hashing and
```

(*continued on next page*)

```
    block-wise histograms) (lines 14–17) */
15: Transform the $T_2$ outputs in $\chi_{\mathrm{i}}^{\mathrm{t}}$ back into a single integer value: $\varphi_{\mathrm{i}}^{\mathrm{t}} = \sum_{t=1}^{T_2} 2^{\mathrm{t}-1} H\left(Z_{\mathrm{i}}^{\mathrm{t}} * \mathbf{J}_t^2\right)$;
16: Calculate the histogram ($2^{T_2}$ bins) of the decimal values on each block;
17: Concatenate all $L$ histograms into one vector: $Lh(\varphi_{\mathrm{i}}^{\mathrm{t}}), t = 1, 2, \ldots, T_1$;
18: Employ the SVM classifier to predict whether each instance module for the target project is defective;
19: Achieve the final prediction result $R_H$;
20: return $R_H$.
```

The solution of this problem can be achieved by the classical principal component analysis, i.e., the first n eigenvectors of the covariance matrix of the matrix $\mathbf{Z}$. Therefore, the corresponding PCA filter (line 7) is expressed as follows:

$$\mathbf{J}_t^1 = \mathrm{M}_{c_1,c_2}\left(\mathbf{u_t}\left(\mathbf{ZZ^T}\right)\right), t = 1, 2, \ldots, T_1, \tag{12}$$

where $\mathbf{u_t}\left(\mathbf{ZZ}^T\right)$ denotes the tth principal eigenvector of $\mathbf{ZZ}^T$. M_{c_1,c_2} represents a function that maps $\mathbf{u_t}\left(\mathbf{ZZ^T}\right) \in \mathbb{R}^{c_1 c_2}$ to a matrix $\mathbf{J} \in \mathbb{R}^{c_1 \times c_2}$. This Eq. (12) aims to extract the eigenvectors corresponding to the first T_1 maximum eigenvalues of the covariance matrix to form the eigenmapping matrix. The principal components retain the main information of the zero-mean training instances.

In the second stage (PCA), the process is basically the same as the mapping mechanism of the first stage. We first convolve N' original instances with the PCA filters learned in the first stage to obtain $N' * T_1$ instances, so the mapping output of the tth filter in the first stage is calculated as follows: $Z_{\mathrm{i}}^{\mathrm{t}} = Z_{\mathrm{i}} * \mathrm{J}_{\mathrm{t}}^1,\ i = 1, 2, \ldots, N'$, where * deNotes 2D convolution (line 8). Since the convolution operation will result in a smaller size, it is necessary to implement the zero-padded operation on defect instances before calculating convolution mapping, so as to ensure that the mapping result is the same size as the original defect instance matrix.

Like the first stage, the input matrix (i.e., the mapping output of the first stage) also needs to employ the $c_1 \times c_2$ filter to extract all defect metrics, remove mean and concatenate in the second stage, and then we can achieve the new training instance matrix $\mathbf{H}$ (lines 9–11). Similarly, the PCA filters in the second stage are also formed by selecting the eigenvectors corresponding to the covariance matrix (line 12):

$$\mathbf{J}_t^2 = \mathrm{M}_{c_1,c_2}\left(\mathbf{u_t}\left(\mathbf{HH^T}\right)\right), t = 1, 2, \ldots, T_2. \tag{13}$$

Next we convolve $N' * T_1$ instances with the PCA filters learned in the second stage to obtain $N' * T_1 * T_2$ instances, so the mapping output of the tth filter in the second stage (line 13) is calculated as follows:

$$\chi_{\mathrm{i}}^{\mathrm{t}} = \left\{Z_{\mathrm{i}}^{\mathrm{t}} * \mathbf{J}_t^2\right\}_{\mathrm{t}=1}^{\mathrm{T}_2}. \tag{14}$$

In the output stage (binary hashing and block-wise histograms), each of the $N' * T_1$ instances obtained in the first stage corresponds to the T_2 output instances obtained in the second stage (line 14). We first binarize the T_2 instances $\left\{Z_{\mathrm{i}}^{\mathrm{t}} * \mathbf{J}_t^2\right\}_{\mathrm{t}=1}^{\mathrm{T}_2}$ in the second stage. Then we perform hash encoding on these binarized features, i.e., encoding the T_2 instances for each output instance in the first stage, and we can obtain $N' * T_1$ instances. We can transform the T_2 outputs in $\chi_{\mathrm{i}}^{\mathrm{t}}$ back into a single integer value in the range $\left[0, 2^{T_2} - 1\right]$ (line 15), as shown in Eq. (15):

$$\varphi_{\mathrm{i}}^{\mathrm{t}} = \sum_{t=1}^{T_2} 2^{\mathrm{t}-1} H\left(Z_{\mathrm{i}}^{\mathrm{t}} * \mathbf{J}_t^2\right), \tag{15}$$

where $H(.)$ denotes a Heaviside step function, whose value is 1 for positive entries and 0 otherwise.

Each of the T_1 instances $\varphi_{\mathrm{i}}^{\mathrm{t}}, t = 1, \ldots, T_1$ is divided into L blocks. We calculate the histogram (2^{T_2} bins) of the decimal values on each block (line 16), and then concatenate all L histograms into one vector, which can be represented as $Lh(\varphi_{\mathrm{i}}^{\mathrm{t}}), t = 1, 2, \ldots, T_1$ (line 17), thereby completing the metric extraction of defect instances by PCANet. Finally, we employ the SVM classifier to predict whether each instance module for the target project is defective (line 18).

Similar to the traditional convolutional neural network, the network parameters, including the filter size, the number of filters and the number of layers need to be set in advance. Once these network parameters are fixed, the training of the PCANet is relatively easy since the filter learning in the PCANet does not refer to regularization parameters and complex optimization procedure.

The time complexity of the PCANet predictor includes two parts: the time complexity of PCANet and the time complexity of SVM. The time complexity of PCANet is calculated as $O\left(pqc_1c_2(T_1 + T_2) + pq(c_1c_2)^2\right)$, where p, q denote the width and height of the matrix reshaped for each defect instance, c_1, c_2 denote the filter size, and T_1, T_2 denote the number of filters in each stage. The time complexity of SVM is calculated as $O\left(dN^2\right)$, where d denotes the metric dimension, and N denotes the number of training instances. Therefore, the time complexity of PCANet is $O\left(pqc_1c_2(T_1+T_2)+pq(c_1c_2)^2\right)+O\left(dN^2\right)$.

4. Experimental setup

In this section, we show the experimental setup, including experimental subjects, evaluation indicators, experimental design, parameter settings, and statistical significance test.

4.1. Experimental subjects

We conduct extensive experiments on 26 commonly used open-source projects from five datasets, including PROMISE[1] [27], ReLink[2] [55], NASA[1] [38], AEEEM[3] [9] and SOFTLAB[1] [46]. Table 1 details the statistics for 26 software projects. We choose 8 software projects from the PROMISE dataset, which are collected by Turhan et al. [27]. Each project has 20 metrics in the PROMISE dataset, which consists of Object-Oriented (OO) metrics, McCabe's cyclomatic metrics, CK metrics. The ReLink dataset is collected by Wu et al. [55], which consists of three software projects: Apache, Safe, ZXing, and these projects have 26 defect metrics. For the NASA dataset, we select 5 software projects with different metric sets as experimental subjects. The AEEEM dataset is collected by D'Ambros et al. [9], which contains 61 metrics: 5 entropy-of-change metrics, 5 previous-defect metrics, 17 entropy-of-source-code metrics, 17 source code metrics, and 17 churn-of-source code metrics. The SOFTLAB dataset contains 5 versions of the AR project with 29 metrics. The granularity represents the granularity level of defect instances that need to be predicted. Since heterogeneous defect prediction at different granularity levels can find appropriate training data for building the prediction model, this type of prediction model is more universal and practical. In addition, we also find that each project in the five datasets is class imbalanced, among which the imbalanced rates of camel-1.0, ZXing, PC2, LC and AR1 in PROMISE, Relink, NASA, AEEEM and SOFTLAB reach the highest 25.08, 2.38, 45.56, 9.80 and 12.44 respectively. Therefore, it is very necessary to perform class imbalance processing on them when these projects are used as source projects.

Since we focus on predictions across different datasets with heterogeneous metric sets, 542 possible prediction combinations from 26 software projects are conducted for heterogeneous defect prediction in our IVKMP model. For example, when the ant-1.3 project in the PROMISE dataset is used as the target project, we need to build the HDP model using 18 projects in the other four datasets besides the 8 projects in the PROMISE dataset as the source project respectively (a total of 18 prediction combinations), so as to ensure that the data distribution between the source project and the target project is heterogeneous. In addition, some NASA projects do not have the same metric sets, e.g., between MW1 (37) and MC2 (39), PC2 (36). When the MW1 project in the NASA dataset is used as the target project, we also conduct heterogeneous defect prediction on prediction combinations of NASA projects with different metric sets (i.e., MC2 $\mapsto$ MW1, PC2 $\mapsto$ MW1), not just the combinations of 21 projects in the other four datasets.

4.2. Evaluation indicators

We employ some evaluation indicators to evaluate the performance of the IVKMP model, including F1, G-measure, MCC, AUC, HV and running time, in which F1, G-measure, MCC and AUC are commonly used in many software defect prediction studies [42,47,65,15,34,69]. F1, G-measure and MCC can be directly calculated by a confusion matrix, and this matrix contains four possible classification results, i.e., TP, FP, FN, and TN, as shown in Table 2.

F1: An comprehensive measure for harmonic mean of precision and recall.

$$F1=\frac{2\times precision\times recall}{precision+recall} \tag{16}$$

$G-$**measure**: An comprehensive measure for harmonic mean of pd and 1-pf.

$$G-measure=\frac{2\times pd\times(1-pf)}{pd+(1-pf)} \tag{17}$$

MCC (**Matthews Correlation Coefficient**): The correlation between the true and predicted binary classification, which is a comprehensive measure of four possible classification results.

$$MCC=\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{18}$$

1 http://openscience.us/repo/.
2 http://www.cse.ust.hk/scc/ReLink.htm.
3 http://bug.inf.usi.ch/.

Table 1
Statistics for 26 software projects.

Datasets	*Granularity*	*Projects*	*#of metrics*	*#of instances*	*#of defective instances*	*#of non−defective instances*	*Defective ratio(%)*	*Imbalance ratio*
PROMISE	Class	ant-1.3	20	125	20	105	16.00	5.25
		camel-1.0	20	339	13	326	3.83	25.08
		poi-1.5	20	237	141	96	59.49	0.68
		redaktor	20	176	27	149	15.34	5.52
		xalan-2.4	20	723	110	613	15.21	5.57
		xerces-1.2	20	440	71	369	16.14	5.20
		synapse-1.2	20	256	86	170	33.59	1.98
		lo4j-1.0	20	135	34	101	25.19	2.97
ReLink	File	Apache	26	194	98	96	50.52	0.98
		Safe	26	56	22	34	39.29	1.55
		ZXing	26	399	118	281	29.57	2.38
NASA	Function	MW1	37	264	26	238	9.85	9.15
		MC2	39	125	44	81	35.20	1.84
		PC1	37	759	61	698	8.04	11.44
		PC2	36	745	16	729	2.15	45.56
		PC4	37	1458	178	1280	12.21	7.19
AEEEM	Class	JDT	61	997	206	791	20.66	3.84
		LC	61	691	64	627	9.26	9.80
		EQ	61	324	129	195	39.81	1.51
		PDE	61	1492	209	1283	14.01	6.14
		ML	61	1862	245	1617	13.16	6.60
SOFTLAB	Function	AR1	29	121	9	112	7.44	12.44
		AR3	29	63	8	55	12.70	6.88
		AR4	29	107	20	87	18.69	4.35
		AR5	29	36	8	28	22.22	3.50
		AR6	29	101	15	86	14.85	5.73

Table 2
Confusion matrix.

	Positive (Predicted)		Negative (Predicted)
True (Actual)	TP		FN
Flase (Actual)	FP		TN
precision		$\frac{TP}{TP+FP}$	
recall or pd		$\frac{TP}{TP+FN}$	
pf		$\frac{FP}{FP+TN}$	

AUC: AUC represents the **A**rea **U**nder the *R*eceiver *O*perating *C*haracteristic (*ROC*) **C**urve (**AUC**), and the curve can be drawn in a 2D space with pf (probability of false alarm) as x-axis and pd (probability of detection) as y-axis. AUC is independent from the cutoff threshold used to decide whether an instance is classified as positive or negative.

The larger these four indicator values, the better the prediction performance.

HV(**HyperVolume**): We use the HV indicator to measure the convergence and diversity of multi-objective metric subset selection optimization approaches, so as to evaluate the quality of the selected solutions. The larger the HV value, the better the performance.

We also employ the running time to measure the efficiency of the multi-objective VaEA optimization approach and the PCANet predictor in IVKMP.

4.3. Experimental design

To comprehensively demonstrate the validity of the proposed data-driven IVKMP model, we design the following experiments.

(1) Performance and efficiency verification for multi-objective VaEA optimization approach

(a) Performance verification for VaEA.

To verify the effect of the metrics selected by VaEA in IVKMP on HDP model performance, we compare VaEA with eight state-of-the-art multi-objective metric subset selection approaches, including hpaEA (**h**yper**p**lane **a**ssisted **E**volutionary

Algorithm) [3], KnEA (**Kn**ee point-driven **E**volutionary **A**lgorithm) [67], MaOEA-R&D (**Ma**ny-**O**bjective **E**volutionary **A**lgorithm – objective space **R**eduction and **D**iversity improvement) [22], MO-CMA (**M**ulti-**O**bjective **C**ovariance **M**atrix **A**daptation) [25], MOEA-D-DU (**M**ulti-**O**bjective **E**volutionary **A**lgorithm – **D**ecomposition – **D**istance-based **U**pdating) [63], NSLS (**N**ondominated **S**orting and **L**ocal **S**earch) [2], PICEA-g (**P**reference-**I**nspired **C**o-**E**volutionary **A**lgorithm – **g**oals) [48], RM-MEDA (**R**egularity **M**odel-based **M**ulti-objective **E**stimation of **D**istribution **A**lgorithm) [66]. For a fair comparison, we replace VaEA with eight multi-objective metric subset selection approaches respectively in our IVKMP model, so as to more clearly observe the impact of each approach on HDP performance.

(b) Efficiency verification for VaEA.

Efficiency is an important evaluation indicator to reflect the practicability of metric selection approaches because it is closely related to the cost of software development. Therefore, we need to compare the running time of VaEA with that of eight multi-objective metric selection approaches.

(c) More analysis.

The multi-objective VaEA algorithm mainly considers two optimization objectives into the metric selection problem in software defect prediction. One objective is to minimize the number of selected defect metrics, which is concerned with the cost of the constructed model. Another objective is to minimize the error of the model, which involves in the benefit of the model. Therefore, we need to record the minimum number of selected metrics and the minimum error on each source project.

Convergence and diversity are two important factors that reflect the quality of the solutions achieved by multi-objective metric selection algorithms, and the HV indicator can measure the convergence and diversity of multi-objective metric selection algorithms. Therefore, this experiment is designed to measure the convergence and diversity of the multi-objective VaEA optimization approach compared to eight state-of-the-art multi-objective metric subset selection approaches on each source project in terms of HV.

Furthermore, since the value of parameter k in KNN will affect the performance of the multi-objective metric selection approach, we need to adjust and set the k value via specific experimental verification.

(2) Performance and efficiency verification for the PCANet predictor

(a) Performance verification for PCANet.

In our IVKMP model, we employ a lightweight deep learning network – PCANet with SVM to construct a powerful defect predictor after metric matching and transfer, in which the PCANet is capable of essentially capturing semantically related robust representations for defect metrics and the SVM classifier can predict whether each instance module is defective. To verify the discriminating capacity of the PCANet predictor, we compare the PCANet predictor with seven classic defect predictors, including KNN (K-Nearest Neighbor), NB (Naïve Bayes), SVM (Support Vector Machine) without PCANet, DT (Decision Tree), RF (Random Forest), LR (Logistic Regression) and MLP (Multi-Layer Perceptron). For this experiment, we only replace the PCANet predictor in our IVKMP model with these seven classic defect predictors respectively.

(b) Efficiency verification for PCANet.

To comprehensively evaluate our PCANet predictor in IVKMP, we also compare it with seven classic defect predictors in terms of efficiency, so as to verify whether the PCANet predictor has advantages or whether the running time is within an acceptable range.

(3) Generalization capabilities verification of VaEA and PCANet

The stronger the generalization capability of a model, the lower the probability of overfitting and the better the model performance, and this model can adapt to more complex data distribution or more kinds of datasets. Therefore, the generalization capability can largely reflect the performance of a model. The experiment is designed to demonstrate the generalization capabilities of the multi-objective VaEA optimization approach and PCANet predictor in IVKMP. We pair eight baseline multi-objective metric subset selection approaches and seven classic baseline defect predictors, and compare the HDP performance of multi-objective VaEA optimization approach on the PCANet predictor with these paired models. Note that these paired models also adopt InfoGAN and feature matching and transfer in this experiment.

(4) Performance verification for deep generation InfoGAN model

(a) Influence for InfoGAN on the proposed IVKMP model.

In this paper, we utilize an advanced deep generation model – InfoGAN to conduct data augmentation tasks for source software projects, including generating more defect instances and achieving class balance. To test whether or how much InfoGAN would affect the performance of our IVKMP model, we compare IVKMP using InfoGAN with IVKMP without using InfoGAN.

(b) Three types of loss values for InfoGAN on each source project.

In this experiment, to more directly reflect the quality of the generated defect instances, we explore the three types of loss values for generating defect instances on each source software project using the deep generation model InfoGAN, including discriminator loss, recognition loss and generator loss.

(c) Running time for InfoGAN on each source project.

To more comprehensively investigate the efficiency of InfoGAN, we also provide the running time for generating defect instances on 26 source software projects using the InfoGAN.

(5) Superiority verification for the proposed IVKMP model

Previous studies have proposed some classic HDP models, including FMT [61], HDP-KS [35], CPDP-IFS [20], and CCA+ [26]. To verify the superiority of our IVKMP model, we compare it with these four HDP models in prediction performance.

4.4. Parameter settings

In this section, we adjust and set some important parameters for our IVKMP model and baseline models.

In InfoGAN, some important parameters of the network structure need to be adjusted or set. For the generator, the kernel size is set to 3 and the number of filters is set to 64 in the first two convolutional layers, and the kernel size is set to 2 and the number of filters is set to 64 in the third convolutional layer. Moreover, in the first two convolutional layers, the nonlinear activation function all adopts "*relu*", and the momentum in BatchNormalization is set to 0.8; in the third convolutional layer, the nonlinear activation function adopts "*tanh*". For the shared layers between discriminator and recognition network, the kernel size is set to 3 and the number of filters is set to 64 in the first convolutional layer, and the kernel size is set to 3 and the number of filters is set to 128 in the second convolutional layer, and the kernel size is set to 3 and the number of filters is set to 256 in the third and fourth convolutional layer. In four convolutional layers, the nonlinear activation function all adopts the "*LeakyReLU*" (*alpha* = 0.2), and the dropout rate is set to 0.25, and the momentum in BatchNormalization is set to 0.8.

Among the nine multi-objective metric subset selection optimization approaches, there are some common parameters that need to be set, such as the number of objectives: 2 (i.e., the error and the number of selected metrics), the population size: 100, the number of evaluations: 10000, and the number of decision variables (*DV*) is set to the number of metrics on each source project, and the individual encoding adopts the binary encoding strategy. The parameter *k* for KNN is set to 3 in all multi-objective approaches. Moreover, there are also some special parameters that need to be set. For KnEA, the rate of knee points in the population is set to 0.5, and the mating selection adopts the binary tournament selection mechanism. For MOEAD-DU, the probability *delta* of choosing parents locally is set to 0.9, and the number of nearest weight vectors is set to 5. For NSLS, the mean value of the Gaussian distribution is set to 0.5, and the standard deviation *delta* of the Gaussian distribution is set to 0.1. For RM-MEDA, the number of clusters is set to 5.

For the eight defect predictors, some important parameters need to be adjusted or set. For the PCANet predictor, the number of PCA stages is 2, the filter size c_1, c_2 are set to 2, 2 respectively, the number of filters T_1, T_2 in each stage are set to 2, 2 respectively, and the block size for local histograms in the output layer are set to 2, 2. p, q denote the width and height of the matrix reshaped for each defect instance, and their size varies with the dimension of the defect instances after metric matching and transfer. For example, for a 9-dimensional defect instance after metric matching and transfer, p, q are set to 3, 3. For KNN, the number of neighbors is set to 3. For NB, we adopt the kernel estimator that achieves better AUC values on most heterogeneous project pairs. For SVM, we choose the Gaussian RBF as the kernel function. As mentioned by Hsu and Lin [23], the cost parameter $C = 2^{-2}, 2^{-1}, \ldots, 2^{12}$ while the kernel parameter $gamma = 2^{-10}, 2^{-9}, \ldots, 2^{4}$. The optimal parameter combination for the cost parameter *C* and the kernel parameter *gamma* can be obtained according to the optimal AUC value by the grid search. For DT, the criterion adopts *gini*, and the minimum sample leaf is set to 1, and the minimum sample split is set to 2. For RF, the number of variables for random metric selection is set to 2, and the number of generated trees is set to 8, and the criterion adopts *gini*. For LR, the distribution adopts *normal* and the random state adopts *None*, and the tolerant error is set to 1E−4. For MLP, we adopt a neural network with two hidden layers, and set the number of hidden nodes from 5 to 100 with an increment of 5, and the optimal number of hidden nodes is determined based on the best AUC by the grid search. The tolerant error is set to 1E−4, and the learning rate is set to 1E−3.

4.5. Statistical significance test

In this paper, we first employ the Scott-Knott Effect Size Difference (ESD) test [43,44] to rank multiple metric subset selection approaches and multiple defect predictors in terms of four indicators. Then we exhibit the boxplots with the Scott-Knott ESD test to visual the performance differences for these metric subset selection approaches and defect predictors. Each color denotes a rank, and there is a statistically significant performance difference between the two approaches in different ranks. The Scott-Knott Effect Size Difference (ESD) test is an alternative approach of the Scott-Knott test, which is a mean comparison approach that utilizes the hierarchical clustering to partition multiple approaches into statistically distinct groups with non-negligible difference, and considers the magnitude of the difference (i.e., effect size) for multiple approaches within a group and between groups.

We also utilize Wilcoxon signed-rank test [10] and Cliff's Delta effect size analysis [30] to check whether the models are statistically significant. The Wilcoxon signed-rank test [10] can be used for measuring the difference between data pairs, and does not demand the underlying software metrics to follow any data distribution. At the 95% confidence level, p-values that are less than 0.05 indicate that the differences between approaches are statistically significant. To test the effectiveness between our approaches and the baselines, we further leverage Cliff's delta (δ) [30] to measure the effect size between them. The Cliff's delta (δ) represents a non-parametric effect size evaluation that tests the difference degree between two approaches. The δ is a evaluation of how often the values in one approach are larger than the values in the other approach. The δ value is in the interval [-1,1], where δ = 1 or -1 denotes that all values in one approach are larger or smaller than those of the other approach, and 0 denotes that all values in the two approaches are completely overlapping. Table 3 shows the different δ values and their corresponding effectiveness levels.

Table 3
Cliff's Delta and the effectiveness level.

Cliff's Delta ($\|\delta\|$)	Effectiveness Level
$\|\delta\| < 0.147$	Negligible (N)
$0.147 \leqslant \|\delta\| < 0.33$	Small (S)
$0.33 \leqslant \|\delta\| < 0.474$	Medium (M)
$0.474 \leqslant \|\delta\|$	Large (L)

5. Experimental results

In this section, we report and analyze the experiment result for our IVKMP model.

5.1. Performance and efficiency verification for multi-objective VaEA optimization approach

We first explore the performance and efficiency for VaEA compared with eight state-of-the-art multi-objective metric subset selection approaches, and then report that what is the minimum number of selected metrics while achieving the minimum error on each source project. Finally, we employ the HV indicator to evaluate the convergence and diversity of all nine multi-objective optimization approaches, and provide the setting strategy of k value in KNN.

5.1.1. Performance verification for VaEA

In this experiment, we compare our multi-objective VaEA optimization approach with eight state-of-the-art multi-objective metric subset selection approaches across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC, including hpaEA, KnEA, MaOEA-R&D, MO-CMA, MOEA-D-DU, NSLS, PICEA-g, RM-MEDA. Since our IVKMP model uses VaEA as the metric selection approach, the HDP performance achieved by VaEA is expressed by IVKMP in Table 4 and Fig. 2.

Table 4 exhibits the F1, G-measure, MCC and AUC of all nine multi-objective metric subset selection approaches across total heterogeneous project pairs of 26 target projects from five datasets, including 8 projects from PROMISE, 3 projects from ReLink, 5 projects from NASA, 5 projects from AEEEM, 5 projects from SOFTLAB. In Table 4, we record the average HDP performance indicators of all projects on each dataset (PROMISE, ReLink, NASA, AEEEM, SOFTLAB), and the average HDP performance indicators of 26 projects in all five datasets (ALL). Note that the highest value of each row is denoted in bold. From Table 4, we can find that the VaEA in IVKMP can achieve the optimal average HDP performance in terms of four indicators except for the G-measure on ReLink and the MCC on NASA. More specifically, for all five datasets (ALL), the average F1 (0.5394) by VaEA obtains improvements between 10.60% (for PICEA-g) and 40.69% (for RM-MEDA) with an average improvement of 25.77%, the average G-measure (0.6967) by VaEA yields improvements between 6.74% (for hpaEA) and 21.02% (for

Table 4
Four average indicator values for VaEA in IVKMP compared with eight baseline multi-objective metric subset selection approaches.

Datasets	*Indicators*	*hpaEA*	*KnEA*	*MaOEA – R&D*	*MO – CMA*	*MOEA – D – DU*	*NSLS*	*PICEA – g*	*RM – MEDA*	*IVKMP*
PROMISE	F1	0.5039	0.4823	0.4416	0.4109	0.5304	0.4172	0.5583	0.3964	**0.6310**
	G-measure	0.6288	0.6150	0.6033	0.5777	0.6165	0.5797	0.6104	0.5663	**0.6437**
	MCC	0.2869	0.2817	0.2352	0.2257	0.3063	0.2260	0.3286	0.2213	**0.3606**
	AUC	0.7173	0.7086	0.7083	0.6475	0.7324	0.6730	0.7227	0.6488	**0.7387**
ReLink	F1	0.5383	0.5177	0.4953	0.5391	0.5486	0.5099	0.5680	0.5123	**0.6409**
	G-measure	0.6439	0.6309	0.6057	0.5614	0.6679	0.6380	**0.6824**	0.6095	0.6723
	MCC	0.2713	0.2372	0.2455	0.2215	0.2893	0.2333	0.3112	0.2291	**0.3459**
	AUC	0.6781	0.6633	0.6854	0.6504	0.6938	0.6674	0.6862	0.6585	**0.7236**
NASA	F1	0.3960	0.3811	0.3608	0.3398	0.3800	0.3513	0.3934	0.3260	**0.4014**
	G-measure	0.6573	0.6101	0.6137	0.5583	0.6346	0.6057	0.6431	0.5645	**0.7065**
	MCC	0.2527	0.2587	0.2613	0.2052	**0.2783**	0.2338	0.2472	0.2213	0.2755
	AUC	0.7476	0.7342	0.7043	0.6278	0.7322	0.6489	0.7331	0.5994	**0.7744**
AEEEM	F1	0.4304	0.4294	0.4087	0.3791	0.4298	0.4003	0.4569	0.3636	**0.5077**
	G-measure	0.6602	0.6379	0.6059	0.5639	0.6692	0.5711	0.6636	0.5451	**0.7181**
	MCC	0.3052	0.2823	0.2954	0.2451	0.3252	0.2722	0.3334	0.2567	**0.3870**
	AUC	0.6839	0.6836	0.6850	0.6329	0.7014	0.6605	0.7230	0.6189	**0.7776**
SOFTLAB	F1	0.3942	0.3825	0.3836	0.4165	0.4346	0.3697	0.4517	0.3622	**0.5016**
	G-measure	0.6839	0.6752	0.6288	0.6338	0.6675	0.5872	0.6942	0.6121	**0.7329**
	MCC	0.3046	0.3057	0.3467	0.3102	0.2975	0.2928	0.3255	0.2810	**0.3789**
	AUC	0.7131	0.6981	0.7229	0.6616	0.7333	0.6819	0.7419	0.6390	**0.7815**
ALL	F1	0.4519	0.4376	0.4148	0.4070	0.4658	0.4028	0.4877	0.3834	**0.5394**
	G-measure	0.6527	0.6319	0.6110	0.5802	0.6459	0.5912	0.6513	0.5757	**0.6967**
	MCC	0.2854	0.2769	0.2744	0.2412	0.3009	0.2501	0.3113	0.2405	**0.3511**
	AUC	0.7114	0.7015	0.7032	0.6439	0.7221	0.6683	0.7242	0.6328	**0.7595**

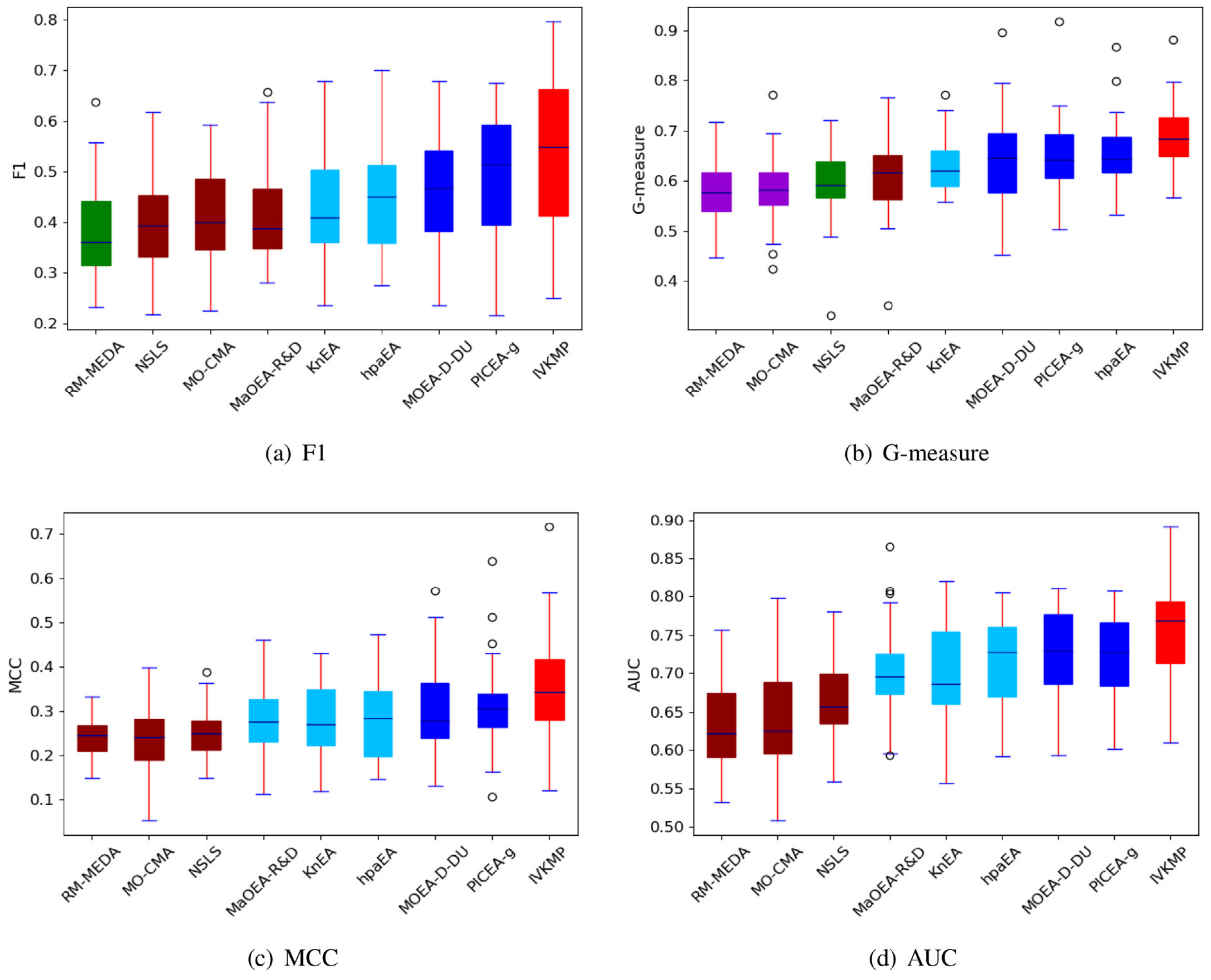


Fig. 2. The Scott-Knott ESD ranking for VaEA in IVKMP compared with eight baseline multi-objective metric subset selection approaches in terms of four indicators.

RM-MEDA) with an average improvement of 13.10%, the average MCC (0.3511) by VaEA achieves improvements between 12.79% (for PICEA-g) and 45.99% (for RM-MEDA) with an average improvement of 29.90%, and the average AUC (0.7595) by VaEA gains improvements between 4.87% (for PICEA-g) and 20.02% (for RM-MEDA) with an average improvement of 10.59% compared with eight state-of-the-art baseline multi-objective metric subset selection approaches.

To perform a visual comparison of the HDP performance differences between the VaEA in IVKMP and eight baseline multi-objective metric subset selection approaches, we depict the boxplots with the Scott-Knott ESD test from the point of four evaluation indicators. Fig. 2 visualizes the differences of the Scott-Knott ESD test for all nine state-of-the-art multi-objective metric subset selection approaches across total 26 target projects in terms of F1, G-measure, MCC and AUC, respectively. Each color denotes a rank: approaches in different ranks have a statistically significant difference in HDP performance. The blue line in each box represents the median indicator value for each multi-objective metric subset selection approach. The x-axis denotes nine multi-objective metric subset selection approaches; the y-axis represents different evaluation indicators. From Fig. 2(a)–(d), we can observe that the VaEA in IVKMP is in the top group (the red box), which indicates that the VaEA can achieve the best HDP performance in terms of four evaluation indicators compared with eight baseline multi-objective metric subset selection approaches. In addition, the median value (the blue line) gained by VaEA is higher than those achieved by eight baseline metric subset selection approaches from the point of four evaluation indicators respectively, which fully demonstrates the superiority of VaEA.

To further explore the impact of VaEA on the performance of the IVKMP model, we utilize the Wilcoxon signed-rank test to validate whether the HDP performance differences between the VaEA and eight baseline metric subset selection approaches are statistically significant. At the 95% confidence level, p-values that are less than 0.05 indicate that the performance differences between different approaches are statistically significant. We also employ Cliff's delta (δ) to evaluate the effect size between the VaEA and eight baseline metric subset selection approaches. Table 5 shows p-values and Cliff's deltas

Table 5
P-value and Cliff's deltas (δ) for VaEA in IVKMP compared with eight baseline multi-objective metric subset selection approaches in terms of four indicators.

Against	F1		G-measure		MCC		AUC	
	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)
hpaEA	0.0002	0.3240(S)	0.0004	0.3254(S)	0.0006	0.3284(S)	0.0003	0.4083(M)
KnEA	0.0001	0.3817(M)	7.84E−05	0.5030(L)	0.0010	0.3669(M)	0.0001	0.4630(M)
MaOEA-R&D	2.35E−05	0.4704(M)	5.68E−05	0.5710(L)	0.0011	0.3861(M)	9.34E−05	0.4867(L)
MO-CMA	2.35E−05	0.4911(L)	1.05E−05	0.7515(L)	2.94E−05	0.5429(L)	1.49E−05	0.7663(L)
MOEA-D-DU	0.0002	0.2766(S)	0.0006	0.3343(M)	0.0009	0.2470(S)	0.0010	0.3077(S)
NSLS	4.10E−05	0.5000(L)	4.10E−05	0.6982(L)	0.0001	0.5178(L)	8.30E−06	0.7189(L)
PICEA-g	0.0002	0.2130(S)	0.0002	0.3580(M)	2.35E−05	0.2396(S)	0.0002	0.3195(S)
RM-MEDA	1.67E−05	0.5666(L)	8.30E−06	0.7840(L)	0.0003	0.6065(L)	8.30E−06	0.8402(L)

(δ) of the VaEA in IVKMP compared with eight baseline metric subset selection approaches in terms of F1, G-measure, MCC and AUC. The meaning of the different Cliff's delta (δ) values and the effectiveness(E) level is shown in Table 3. In terms of four evaluation indicators, the p-values are less than 0.05 (significant) and the δ is larger than 0.147 (not negligible), which is consistent with the observations in Table 4 and Fig. 2.

In conclusion, the multi-objective VaEA optimization approach in IVKMP can achieve the optimal average HDP performance compared with eight state-of-the-art multi-objective metric subset selection approaches across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC.

5.1.2. Efficiency verification for VaEA

The running time can reflect the practicability of a metric selection approach to a great extent, so we compare the running time of VaEA with that of eight multi-objective metric selection approaches. Table 6 records the running time for VaEA in IVKMP compared with eight state-of-the-art multi-objective metric subset selection approaches. We can observe that the VaEA costs the least average running time (171.07) on 26 source software projects, which indicates that the VaEA has the highest selection efficiency. To be specific, the average running time (171.07) by VaEA achieves improvements between 7.38% (for hpaEA) and 40.71% (for MOEA-D-DU) with an average improvement of 24.46% compared with eight baseline multi-objective metric subset selection approaches.

Table 6
Running time for VaEA compared with eight state-of-the-art multi-objective metric subset selection approaches (in seconds).

Projects	*hpaEA*	*KnEA*	*MaOEA − R&D*	*MO − CMA*	*MOEA − D − DU*	*NSLS*	*PICEA − g*	*RM − MEDA*	*VaEA*
ant-1.3	38.85	**38.27**	38.31	57.27	49.97	49.58	39.61	48.10	41.77
camel-1.0	47.32	48.56	55.45	53.32	62.74	95.69	54.61	56.23	**43.68**
poi-1.5	25.15	**23.45**	24.38	30.98	37.26	33.71	25.74	30.43	23.59
redaktor	58.79	**52.35**	52.79	82.04	58.85	88.80	69.65	83.04	66.28
xalan-2.4	162.62	169.06	179.65	198.81	204.85	212.36	201.90	216.00	**160.08**
xerces-1.2	**86.30**	93.71	87.30	103.48	130.83	108.75	110.09	117.08	86.55
synapse-1.2	70.38	76.91	86.71	112.11	123.29	117.31	84.21	107.10	**67.12**
log4j-1.0	32.85	31.85	33.32	40.40	40.99	43.61	32.50	38.07	**30.13**
Apache	37.87	**36.92**	40.27	57.49	56.62	48.91	47.57	57.79	41.55
Safe	8.04	**3.39**	5.16	7.51	8.05	5.48	7.06	6.85	4.44
ZXing	260.67	276.84	317.46	323.72	349.79	299.13	**259.51**	318.28	262.98
MW1	123.79	**121.58**	129.77	173.88	135.27	225.13	141.66	172.85	128.02
MC2	21.18	**20.48**	40.16	31.32	30.29	41.19	22.53	31.72	21.22
PC1	223.91	328.74	364.37	373.28	443.96	425.01	412.30	429.24	**219.25**
PC2	177.45	184.80	193.49	197.29	303.38	236.38	235.38	297.97	**168.07**
PC4	336.56	357.73	378.52	387.73	508.65	432.98	416.53	475.76	**326.04**
JDT	**269.14**	313.11	337.34	350.73	437.05	387.33	368.14	398.18	308.12
LC	157.65	168.99	158.20	218.17	307.37	264.32	245.86	277.14	**153.03**
EQ	123.19	133.23	**123.17**	186.00	135.38	194.39	140.06	195.56	137.46
PDE	**311.02**	388.49	403.23	418.09	543.46	520.76	449.45	483.86	363.26
ML	1976.12	2042.53	1962.61	2530.75	3422.87	2424.76	3786.33	2051.44	**1687.71**
AR1	35.17	39.17	71.18	66.99	**34.63**	55.79	49.15	64.75	47.75
AR3	5.33	**4.99**	6.53	13.32	16.97	8.57	7.35	8.25	5.62
AR4	24.60	27.57	22.79	28.58	25.88	30.03	24.17	27.80	**20.17**
AR5	3.43	**3.07**	3.43	7.68	7.57	5.53	7.16	7.02	4.27
AR6	26.48	29.65	29.23	41.34	**25.38**	40.64	41.56	41.56	29.74
Avg	184.70	192.90	197.88	234.32	288.51	246.01	280.00	232.39	**171.07**

Table 7
The number of selected metrics and classification error achieved by the multi-objective VaEA optimization approach.

Projects	*Selected proportion*	*#of total metrics*	*#of selected metrics*	*Error*	*Projects*	*Selected proportion*	*#of total metrics*	*#of selected metrics*	*Error*
ant-1.3	0.35	20	7	0.11	PC1	0.46	37	17	0.03
camel-1.0	0.35	20	7	0.01	PC2	0.47	36	17	0.01
poi-1.5	0.35	20	7	0.13	PC4	0.22	37	8	0.12
redaktor	0.40	20	8	0.04	JDT	0.16	61	10	0.08
xalan-2.4	0.40	20	8	0.06	LC	0.26	61	16	0.03
xerces-1.2	0.35	20	7	0.05	EQ	0.26	61	16	0.16
synapse-1.2	0.25	20	5	0.11	PDE	0.21	61	13	0.05
lo4j-1.0	0.40	20	8	0.06	ML	0.23	61	14	0.04
Apache	0.31	26	8	0.18	AR1	0.24	29	7	0.04
Safe	0.42	26	11	0.04	AR3	0.38	29	11	0.05
ZXing	0.35	26	9	0.10	AR4	0.38	29	11	0.03
MW1	0.22	37	8	0.06	AR5	0.34	29	10	0.14
MC2	0.31	39	12	0.25	AR6	0.41	29	12	0.01

5.1.3. More analysis

Two objective values: The multi-objective VaEA optimization approach aims to select the fewest representative metrics for the minimum error, so we report that what is the minimum number of selected metrics while achieving the minimum error on each source project. Table 7 shows the project name, the selected proportion, the number of total metrics, the number of selected metrics and the minimum error achieved by multi-objective VaEA algorithm on each source project. From Table 7, we can observe that the selected proportion ranges from 0.16 to 0.47, and the error ranges from 0.01 to 0.25.

HV indicator: We also employ the HV indicator to measure the convergence and diversity of multi-objective VaEA optimization approach compared to eight baseline multi-objective metric subset selection approaches. Table 8 shows the HV for VaEA in IVKMP compared with eight state-of-the-art multi-objective metric subset selection approaches. Note that the highest value of each row is denoted in bold. From Table 8, we can find that the VaEA can achieve the highest average HV value compared with eight baseline multi-objective metric subset selection approaches, which manifests that the VaEA has the best convergence and diversity. To be specific, the average HV (0.9554) by VaEA achieves improvements between 2.13% (for PICEA-g) and 34.98% (for RM-MEDA) with an average improvement of 15.96% compared with eight baseline multi-objective metric subset selection approaches.

The setting of k value in KNN: In general, the parameter k in k-nearest neighbor (KNN) is empirically set to a smaller odd number, namely 3, 5, 7, 9. Therefore, we set the k value to 3, 5, 7, 9 respectively, and perform metric subset selection on each of the 26 software projects. The experimental results show that the multi-objective VaEA algorithm can achieve the minimum error and the minimum number of metrics on most software projects when k is set to 3. To further verify the metric selection capability of VaEA on 26 software projects when k is set to 3,5,7,9, we perform metric matching and transfer on the metrics selected by VaEA in these four cases, and then employ PCANet to conduct defect prediction for each heterogeneous project pair. Fig. 3 shows the average F1 and AUC values of IVKMP when k is set to 3,5,7,9 respectively across 542 heterogeneous project pairs of 26 target projects. Since F1 and AUC are the most important evaluation indicators in software defect prediction, we adopt these two indicators as the ordinate values. From Fig. 3, we can observe that IVKMP can achieve the best average F1 and AUC values when k is set to 3, so the parameter k in KNN is set to 3 in this paper.

5.2. Performance and efficiency verification for the PCANet predictor

In this section, we investigate the performance and efficiency for the PCANet predictor compared with seven classic defect predictors.

Table 8
HV for VaEA in IVKMP compared with eight state-of-the-art multi-objective metric subset selection approaches.

Projects	*hpaEA*	*KnEA*	*MaOEA – R&D*	*MO – CMA*	*MOEA – D – DU*	*NSLS*	*PICEA – g*	*RM – MEDA*	*VaEA*
ant-1.3	0.9393	0.9388	0.7189	0.7422	0.9636	0.7906	0.9567	0.7754	**0.9641**
camel-1.0	0.9048	0.9463	0.9736	0.8104	0.9712	0.7689	0.9712	0.7657	**0.9739**
poi-1.5	0.8877	0.8630	0.8114	0.6950	**0.9141**	0.7225	**0.9141**	0.6705	**0.9141**
redaktor	0.8872	0.9299	0.7991	0.8355	**0.9521**	0.7842	**0.9521**	0.7470	**0.9521**
xalan-2.4	0.9174	0.9174	0.7001	0.7398	0.9422	0.7389	**0.9427**	0.7735	0.9425
xerces-1.2	0.9243	0.9239	0.5726	0.7336	**0.9486**	0.7450	0.9483	0.7359	0.9466
synapse-1.2	0.8932	0.8880	0.6829	0.7764	0.9195	0.7507	0.9231	0.7022	**0.9243**
lo4j-1.0	0.9303	0.9303	0.6658	0.7293	**0.9530**	0.7851	**0.9530**	0.7335	**0.9530**
Apache	0.8493	0.8410	0.7129	0.6442	**0.8937**	0.6331	**0.8937**	0.6188	0.8899
Safe	**0.9882**	**0.9882**	**0.9882**	0.7808	0.9022	0.6818	0.9032	0.6925	**0.9882**
ZXing	0.8891	0.8836	0.6574	0.6971	**0.9443**	0.7069	0.9366	0.6975	0.9376
MW1	0.9057	0.9439	0.7871	0.7014	**0.9652**	0.7070	**0.9652**	0.6822	0.9518
MC2	0.8947	0.8250	0.9088	0.6340	0.9071	0.6603	0.9104	0.5947	**0.9637**
PC1	0.9425	0.9628	0.9410	0.7149	0.9375	0.7429	**0.9782**	0.7167	0.9627
PC2	0.9715	0.9825	0.8458	0.7188	0.8840	0.7197	0.8734	0.7893	**0.9854**
PC4	0.8752	0.8428	0.9506	0.6745	0.8544	0.7017	**0.9538**	0.7000	0.9227
JDT	0.8948	0.8227	**0.9539**	0.6556	0.9490	0.6603	0.9228	0.6697	0.9487
LC	0.9304	0.8848	0.9804	0.6624	0.9436	0.6914	**0.9808**	0.6612	0.9595
EQ	0.9150	0.8676	0.8701	0.6475	**0.9587**	0.6127	0.9481	0.6397	0.9481
PDE	0.9465	0.8937	0.9610	0.6497	0.8127	0.6369	**0.9686**	0.6732	0.9585
ML	0.9519	0.8501	0.9720	0.6400	0.9090	0.6724	0.9016	0.6547	**0.9787**
AR1	0.9363	0.9060	**0.9814**	0.7172	0.9423	0.7166	**0.9814**	0.7470	**0.9814**
AR3	0.9661	0.9661	0.9625	0.7305	**0.9819**	0.7925	0.9812	0.7586	**0.9819**
AR4	0.9678	0.9617	0.9678	0.7381	**0.9864**	0.7385	**0.9864**	0.7385	**0.9864**
AR5	0.9260	0.9260	0.9260	0.6817	**0.9392**	0.7144	**0.9392**	0.7276	**0.9392**
AR6	0.9657	0.9047	0.9549	0.7074	**0.9841**	0.7793	0.7384	0.7384	**0.9841**
Avg	0.9231	0.9073	0.8556	0.7099	0.9331	0.7175	0.9355	0.7078	**0.9554**

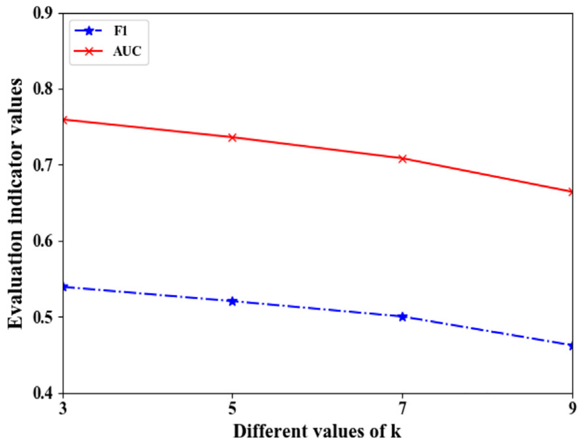


Fig. 3. Average F1 and AUC values of IVKMP with different parameter *k*.

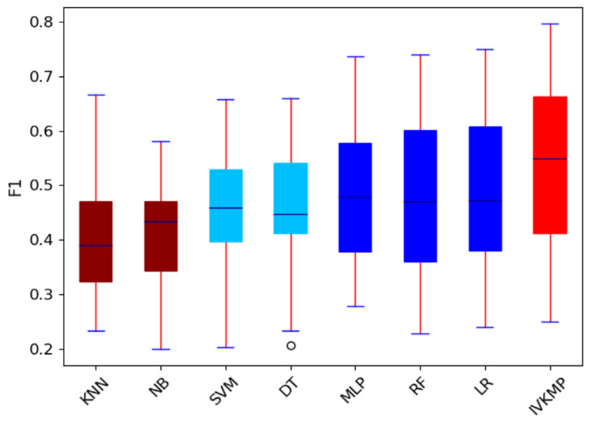
Table 9
Four average indicator values for PCANet in IVKMP compared with seven classic defect predictors.

Datasets	Indicators	KNN	NB	SVM	DT	RF	LR	MLP	IVKMP
PROMISE	F1	0.4089	0.4583	0.5116	0.5414	0.5892	0.5976	0.577	**0.6310**
	G-measure	0.5408	0.5736	0.5890	0.5794	0.5582	0.6049	0.5975	**0.6437**
	MCC	0.2189	0.2451	0.2711	0.2987	0.3265	0.3153	**0.3656**	0.3606
	AUC	0.6505	0.6555	0.6778	0.6201	0.6883	0.6997	0.6665	**0.7387**
ReLink	F1	0.4591	0.4453	0.5083	0.5175	0.5327	0.5457	0.5775	**0.6409**
	G-measure	0.5444	0.5102	0.5728	0.5802	0.5533	0.5807	0.5611	**0.6723**
	MCC	0.2251	0.2689	0.2943	0.2782	0.2907	0.2515	**0.3527**	0.3487
	AUC	0.5535	0.5870	0.6230	0.5860	0.6399	0.6686	0.6977	**0.7236**
NASA	F1	0.3327	0.3615	0.3826	0.3761	0.3670	0.3675	0.3598	**0.4014**
	G-measure	0.6421	0.6853	0.6612	0.7042	0.7003	0.7041	0.7008	**0.7065**
	MCC	0.2313	0.2475	0.2346	0.2723	0.2724	0.2570	**0.3034**	0.2755
	AUC	0.6611	0.6896	0.6810	0.6785	0.7375	0.7183	0.7366	**0.7744**
AEEEM	F1	0.3788	0.4247	0.4531	0.4528	0.4458	0.4666	0.4552	**0.5077**
	G-measure	0.5184	0.5350	0.5935	0.6286	0.5944	0.6664	0.6669	**0.7171**
	MCC	0.2339	0.2726	0.2817	0.3011	0.3427	0.2911	**0.3901**	0.3870
	AUC	0.5934	0.6508	0.6701	0.6180	0.6631	0.7074	0.6155	**0.7776**
SOFTLAB	F1	0.4041	0.3572	0.3778	0.3849	0.4108	0.4266	0.3992	**0.5016**
	G-measure	0.6539	0.6878	0.6675	0.7220	0.7185	0.7186	0.7025	**0.7329**
	MCC	0.3414	0.3164	0.3168	0.3232	0.3727	0.3740	**0.3853**	0.3789
	AUC	0.6829	0.6450	0.7306	0.6975	0.7124	0.7653	0.6812	**0.7815**
ALL	F1	0.3933	0.4123	0.4494	0.4597	0.4781	0.4893	0.4777	**0.5394**
	G-measure	0.5781	0.6023	0.617	0.6404	0.6228	0.6549	0.6467	**0.6967**
	MCC	0.2484	0.2673	0.2776	0.2964	0.3240	0.3034	**0.3607**	0.3511
	AUC	0.6366	0.6512	0.6807	0.6419	0.6919	0.7138	0.6785	**0.7595**

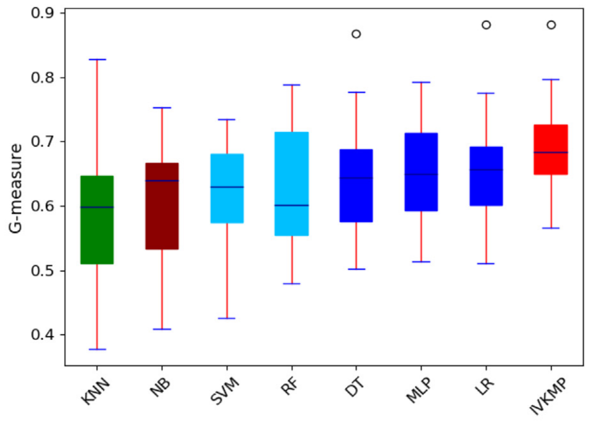
5.2.1. Performance verification for PCANet

We construct a powerful deep defect predictor – PCANet in our IVKMP model after feature matching and transfer, which is capable of essentially capturing semantically related robust representations for defect metrics. Prior studies [49,16,70] have demonstrated that the abstract deep semantic metrics have stronger discriminating capacity for different classes (defective or non-defective). To explore the prediction performance of PCANet, we compare it with seven classic defect predictors across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC, including KNN, NB, SVM, DT, RF, LR and MLP. Since our IVKMP model adopts PCANet as the defect predictor, the HDP performance achieved by PCANet is marked by IVKMP in Table 9 and Fig. 4.

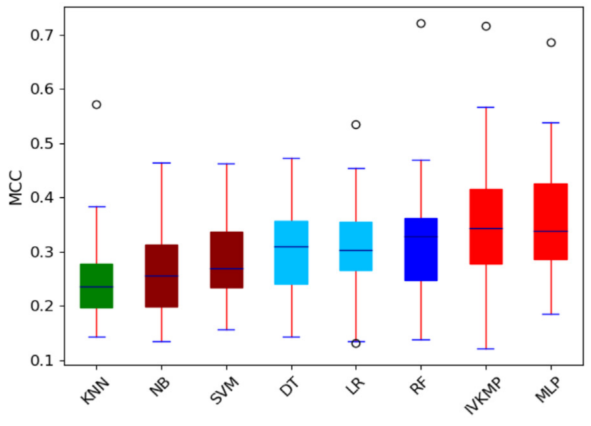
Table 9 shows the F1, G-measure, MCC and AUC of all eight defect predictors across total heterogeneous project pairs of 26 target projects from five datasets, including 8 projects from PROMISE, 3 projects from ReLink, 5 projects from NASA, 5 projects from AEEEM, 5 projects from SOFTLAB. In Table 9, we report four average HDP performance indicators of all projects on each dataset (PROMISE, ReLink, NASA, AEEEM, SOFTLAB), and four average indicators of 26 projects in all five datasets (ALL). Note that the highest value of each row is denoted in bold. From Table 9, we can observe that the PCANet predictor in IVKMP can achieve the best average HDP performance in terms of F1, G-measure and AUC. In terms of MCC, the performance of PCANet is second only to that of MLP. More specifically, for all five datasets (ALL), the average F1 (0.5394) by PCANet yields improvements between 10.24% (for LR) and 37.15% (for KNN) with an average improvement of 20.19%, the average



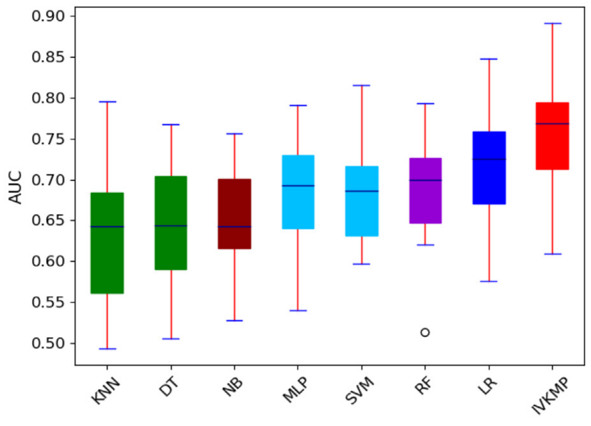
(a) F1



(b) G-measure



(c) MCC



(d) AUC

Fig. 4. The Scott-Knott ESD ranking for PCANet in IVKMP compared with seven classic defect predictors in terms of four indicators.

G-measure (0.6967) by PCANet obtains improvements between 6.38% (for LR) and 20.52% (for KNN) with an average improvement of 11.98%, the average MCC (0.3511) by PCANet gains improvements between −2.70% (for MLP) and 41.34% (for KNN) with an average improvement of 19.86%, and the average AUC (0.7595) by PCANet achieves improvements between 6.40% (for LR) and 19.31% (for KNN) with an average improvement of 13.42% compared with seven classic defect predictors.

To perform a visual comparison of the HDP performance differences between the PCANet in IVKMP and seven classic defect predictors, we depict the boxplots with the Scott-Knott ESD test in terms of four evaluation indicators. Fig. 4 exhibits the differences of the Scott-Knott ESD test for all eight defect predictors across total 26 target projects in terms of F1, G-measure, MCC and AUC, respectively. Each color represents a rank: approaches in different ranks have a statistically significant difference in HDP performance. The blue line in each box denotes the median indicator value for each defect predictor. The x-axis denotes eight defect predictors; the y-axis represents different evaluation indicators. From Fig. 4(a), (b), (d), we can observe that the PCANet in IVKMP is in the highest rank (the red box), which means that the PCANet can boost the HDP performance in terms of F1, G-measure and AUC compared with seven classic defect predictors. In Fig. 4(c), the PCANet in IVKMP and MLP belong to the highest group, which also shows that they can achieve the best prediction performance. In addition, the median value (the blue line) achieved by PCANet is higher than those achieved by seven classic defect predictors from the point of F1, G-measure and AUC respectively, which fully demonstrates the effectiveness of PCANet.

Similarly, we also employ the Wilcoxon signed-rank test to evaluate whether the HDP performance differences between the PCANet and seven baseline defect predictors are statistically significant. At the 95% confidence level, p-values that are less than 0.05 denote that the performance differences between different predictors are statistically significant. We also utilize Cliff's delta (δ) to measure the effect size between the PCANet and seven baseline defect predictors. Table 10 shows p-values and Cliff's deltas (δ) of the PCANet in IVKMP compared with seven baseline defect predictors in terms of F1, G-measure, MCC and AUC. A positive value indicates that the PCANet can boost the HDP prediction performance in terms of

Table 10
P-value and Cliff's deltas (δ) for PCANet in IVKMP compared with seven classic defect predictors in terms of four indicators.

Against	F1		G-measure		MCC		AUC	
	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)
KNN	1.67E−05	0.5385(L)	2.63E−05	0.6923(L)	0.0001	0.5444(L)	1.33E−05	0.7515(L)
NB	2.86E−05	0.4689(M)	0.0004	0.4911(L)	0.0004	0.4497(L)	2.94E−05	0.7751(L)
SVM	0.0001	0.3462(M)	0.0004	0.4438(M)	0.0010	0.3728(M)	3.29E−05	0.6420(L)
DT	9.68E−05	0.3092(S)	0.0005	0.3787(M)	0.0010	0.2515(S)	8.30E−06	0.7692(L)
RF	5.10E−05	0.2308(S)	0.0006	0.4290(M)	0.0215	0.1213(N)	4.57E−05	0.5385(L)
LR	0.0003	0.1982(S)	0.0011	0.2781(S)	0.0031	0.2249(S)	0.0001	0.4024(M)
MLP	0.0003	0.2337(S)	0.0063	0.2485(S)	0.3409	−0.0296	9.68E−05	0.6065(L)

the effect size. In terms of F1, G-measure and AUC, the p-values are less than 0.05 (significant) except for MLP (0.3409) on MCC and the δ is larger than 0.147 (not negligible) except for RF (0.1213) and MLP (-0.0296) on MCC, which is consistent with the observations in Table 9 and Fig. 4.

In brief, the PCANet in IVKMP can enhance the HDP performance compared with seven classic defect predictors across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure and AUC. However, the performance of the PCANet is inferior to that of MLP in terms of MCC.

5.2.2. Efficiency verification for PCANet

We compare the PCANet predictor in IVKMP with seven classic defect predictors in terms of running time, so as to verify whether the PCANet predictor has advantages or whether the running time is within an acceptable range. To more comprehensively investigate the efficiency of PCANet, we count the average running time of all nine multi-objective metric subset selection approaches on each defect predictor.

Table 11 shows the average running time for PCANet compared with seven classic defect predictors. We can observe that our PCANet predictor costs the second longest running time (1.29s) across total heterogeneous project pairs of 26 target projects, but we believe that this time cost is still acceptable, which is considered to sacrifice efficiency for effectiveness.

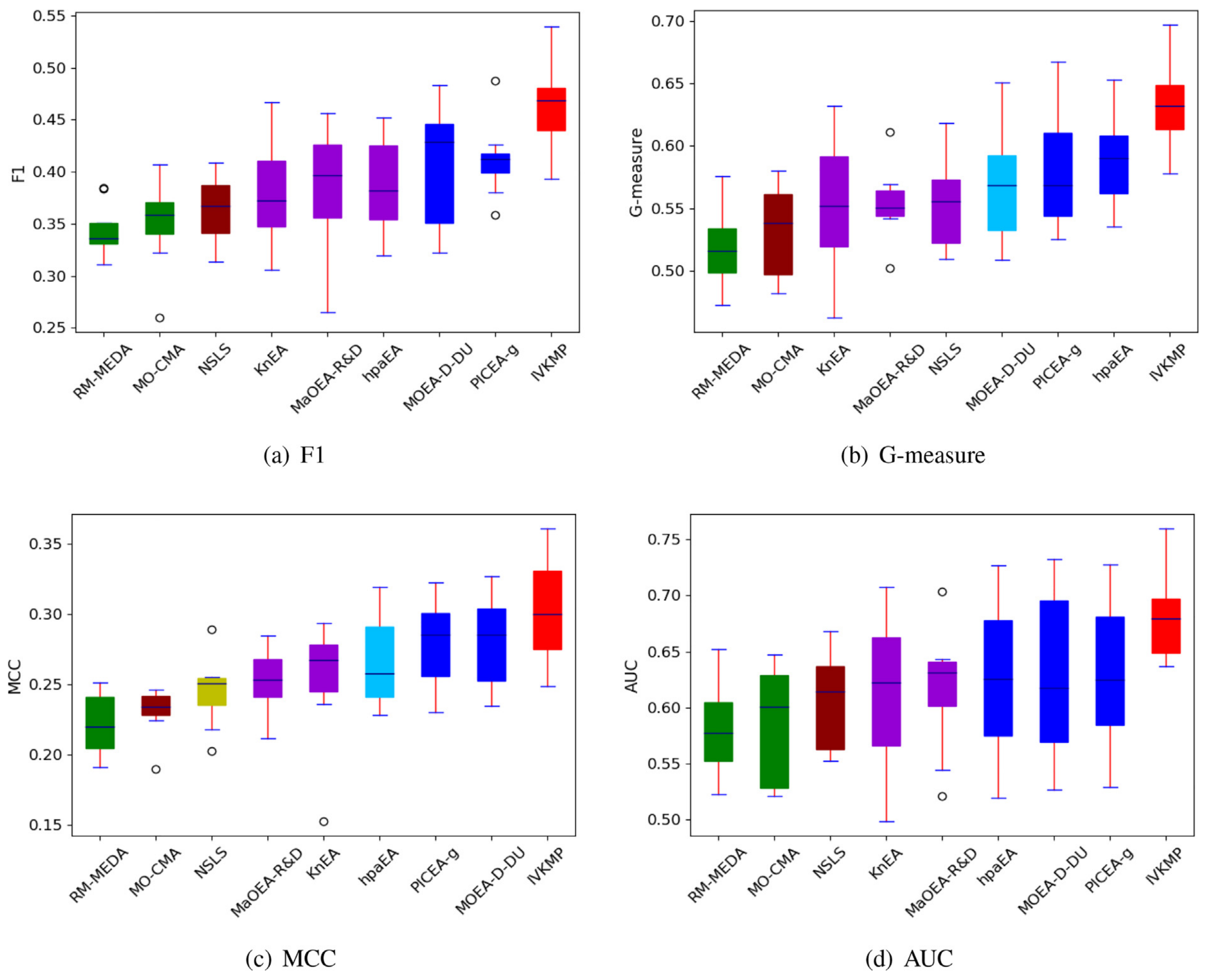
5.3. Generalization capability verification of VaEA and PCANet

In this experiment, we explore the generalization capabilities of the multi-objective VaEA optimization approach and PCANet predictor in IVKMP. In Section 5.1, we compare the multi-objective VaEA optimization approach in IVKMP with eight baseline multi-objective metric subset selection approaches based on the same defect predictor – PCANet. In Section 5.2, we compare the PCANet predictor in IVKMP with seven baseline defect predictors based on the same metric subset selection approach – VaEA. Different from Section 5.1 and Section 5.2, we also pair eight baseline multi-objective metric subset selection approaches and seven classic baseline defect predictors, and compare the HDP performance of multi-objective VaEA optimization approach on the PCANet predictor with these paired models across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC. Since our IVKMP model adopts VaEA as the metric subset selection approach and PCANet as the defect predictor, the performance of VaEA and PCANet is expressed by IVKMP in Figs. 5 and 6.

Figs. 5 and 6 show the boxplots with the Scott-Knott ESD test for validating the generalization capabilities of the multi-objective VaEA optimization approach and PCANet predictor in IVKMP across total 26 target projects in terms of F1, G-measure, MCC and AUC, respectively. In Fig. 5, the blue line in each box denotes the median indicator value for each multi-objective metric subset selection approach. The x-axis represents nine multi-objective metric subset selection approaches; the y-axis denotes different evaluation indicators. We can find that the VaEA in IVKMP is in the top group (the red box), which indicates that the VaEA can achieve the best HDP performance in terms of four evaluation indicators compared with eight baseline multi-objective metric subset selection approaches. In addition, the median value (the blue line) obtained by VaEA is higher than those gained by eight baseline metric subset selection approaches from the point of

Table 11
The average running time for PCANet compared with seven classic defect predictors.

Time	KNN	NB	SVM	DT	RF	LR	MLP	PCANet
hpaEA	0.51	0.50	1.07	0.47	0.55	0.50	3.30	1.39
KnEA	0.27	0.26	0.76	0.26	0.34	0.29	1.92	1.06
MaOEA-R&D	0.33	0.32	1.06	0.32	0.43	0.38	2.48	1.17
MO-CMA	0.32	0.30	0.91	0.31	0.38	0.35	2.10	0.97
MOEA-D-DU	0.33	0.31	0.96	0.33	0.39	0.36	2.18	1.13
NSLS	0.36	0.34	0.98	0.35	0.46	0.39	3.26	1.20
PICEA-g	0.43	0.38	1.21	0.40	0.48	0.44	3.40	1.62
RM-MEDA	0.33	0.34	1.02	0.33	0.42	0.36	2.64	1.87
VaEA	0.39	0.34	1.10	0.35	0.50	0.42	2.84	1.23
Avg	0.36	0.34	1.01	0.35	0.44	0.39	2.68	1.29



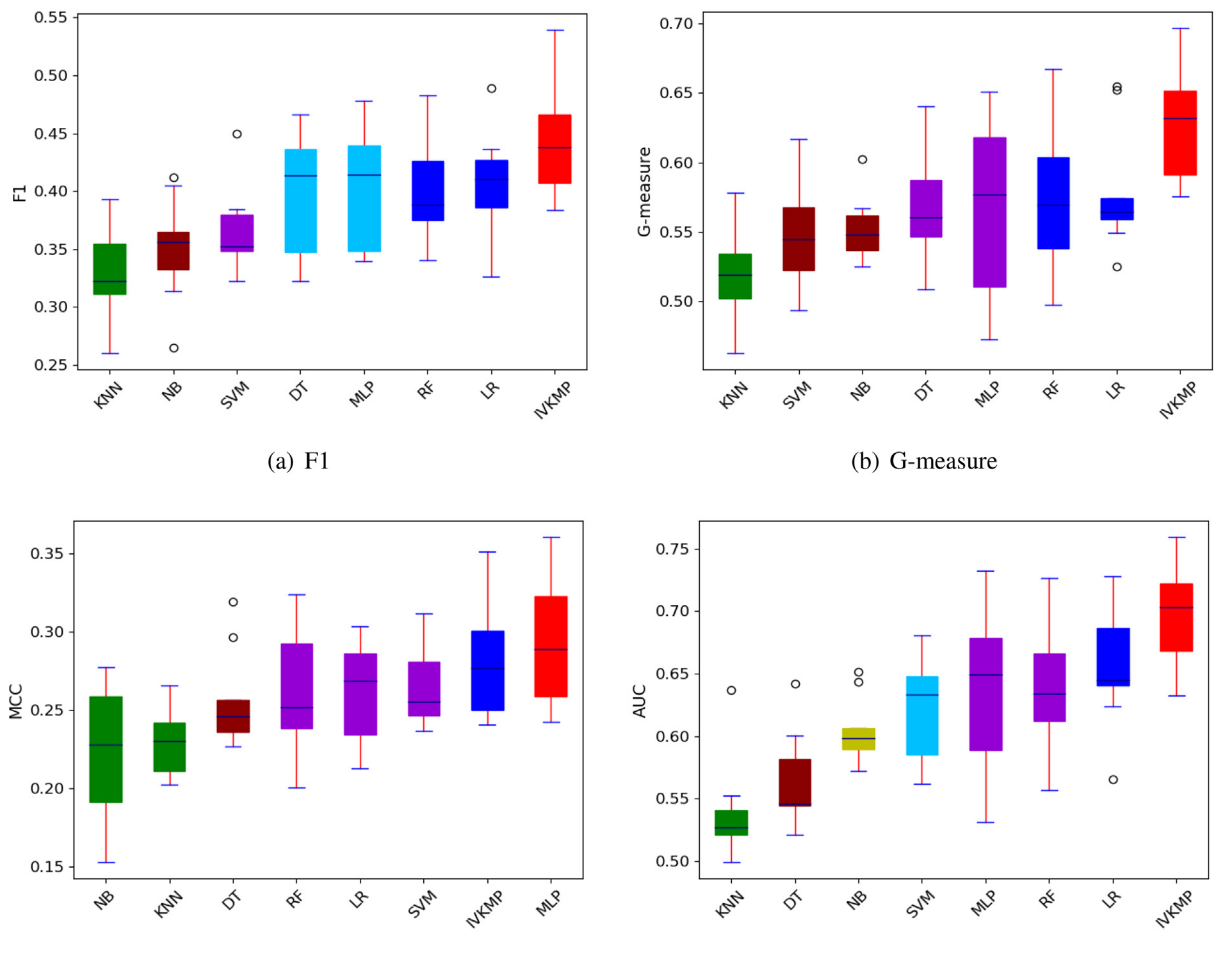
(a) F1 (b) G-measure (c) MCC (d) AUC

Fig. 5. The Scott-Knott ESD ranking for verifying generalization capacity of VaEA in IVKMP in terms of four indicators.

four evaluation indicators respectively. Similarly, in Fig. 6, the blue line in each box represents the median indicator value for each defect predictor. The x-axis denotes eight defect predictor; the y-axis represents different evaluation indicators. From Fig. 6(a), (b), (d), we can observe that the PCANet in IVKMP is in the highest rank (the red box), which indicates that the PCA-Net can enhance the HDP performance in terms of F1, G-measure and AUC compared with seven classic defect predictors. In Fig. 6(c), the performance of PCANet is inferior only to that of MLP. Moreover, the median value (the blue line) achieved by PCANet is higher than those obtained by seven classic defect predictors in terms of F1, G-measure and AUC respectively.

Tables 12, 13 exhibit the p-values and Cliff's deltas (δ) for validating the generalization capabilities of the multi-objective VaEA optimization approach and PCANet predictor in IVKMP across total 26 target projects in terms of F1, G-measure, MCC and AUC, respectively. In Table 12, the p-values are less than 0.05 (significant) except for MOEA-D-DU (0.0929) on MCC in terms of four evaluation indicators, and the δ is larger than 0.147 (not negligible) in terms of four indicators, and even all greater than 0.474 (L) in terms of F1 and G-measure, which is consistent with the observations in Fig. 5. Similarly, in Table 13, in terms of F1, G-measure and AUC, the p-values are less than 0.05 (significant) and even all larger than 0.474 (L) in terms of and G-measure. But the p-values are larger than 0.05 (not significant) on SVM (0.2135), DT(0.0663), LR (0.0506) and MLP (0.0858) in terms of MCC. The δ is larger than 0.147 (not negligible) except for MLP (-0.2099) on MCC in terms of four indicators, and even all greater than 0.474 (L) in terms of G-measure.

To sum up, our multi-objective VaEA optimization approach and PCANet predictor in IVKMP have strong generalization performance in terms of four evaluation indicators compared with eight baseline multi-objective metric subset selection approaches and seven baseline defect predictors.



(a) F1 (b) G-measure

(c) MCC (d) AUC

Fig. 6. The Scott-Knott ESD ranking for verifying generalization capacity of PCANet in IVKMP in terms of four indicators.

Table 12
P-value and Cliff's deltas (δ) for verifying generalization capacity of VaEA in IVKMP in terms of four indicators.

Against	F1		G-measure		MCC		AUC	
	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)
hpaEA	0.0117	0.7813(L)	0.0117	0.5313(L)	0.0251	0.5313(L)	0.0251	0.4375(M)
KnEA	0.0173	0.7813(L)	0.0117	0.7813(L)	0.0499	0.6563(L)	0.0173	0.5625(L)
MaOEA-R&D	0.0117	0.7188(L)	0.0117	0.9375(L)	0.0117	0.7813(L)	0.0117	0.7188(L)
MO-CMA	0.0117	0.9688(L)	0.0117	0.9688(L)	0.0117	1(L)	0.0117	0.8750(L)
MOEA-D-DU	0.0173	0.5000(L)	0.0173	0.6563(L)	0.0929	0.3438(M)	0.0499	0.3125(S)
NSLS	0.0116	0.9375(L)	0.0117	0.8750(L)	0.0117	0.7813(L)	0.0117	0.8438(L)
PICEA-g	0.0117	0.5625(L)	0.0357	0.5625(L)	0.0173	0.3438(M)	0.0173	0.4063(M)
RM-MEDA	0.0117	1(L)	0.0117	1(L)	0.0117	0.9688(L)	0.0117	0.9063(L)

5.4. Performance verification for deep generation InfoGAN model

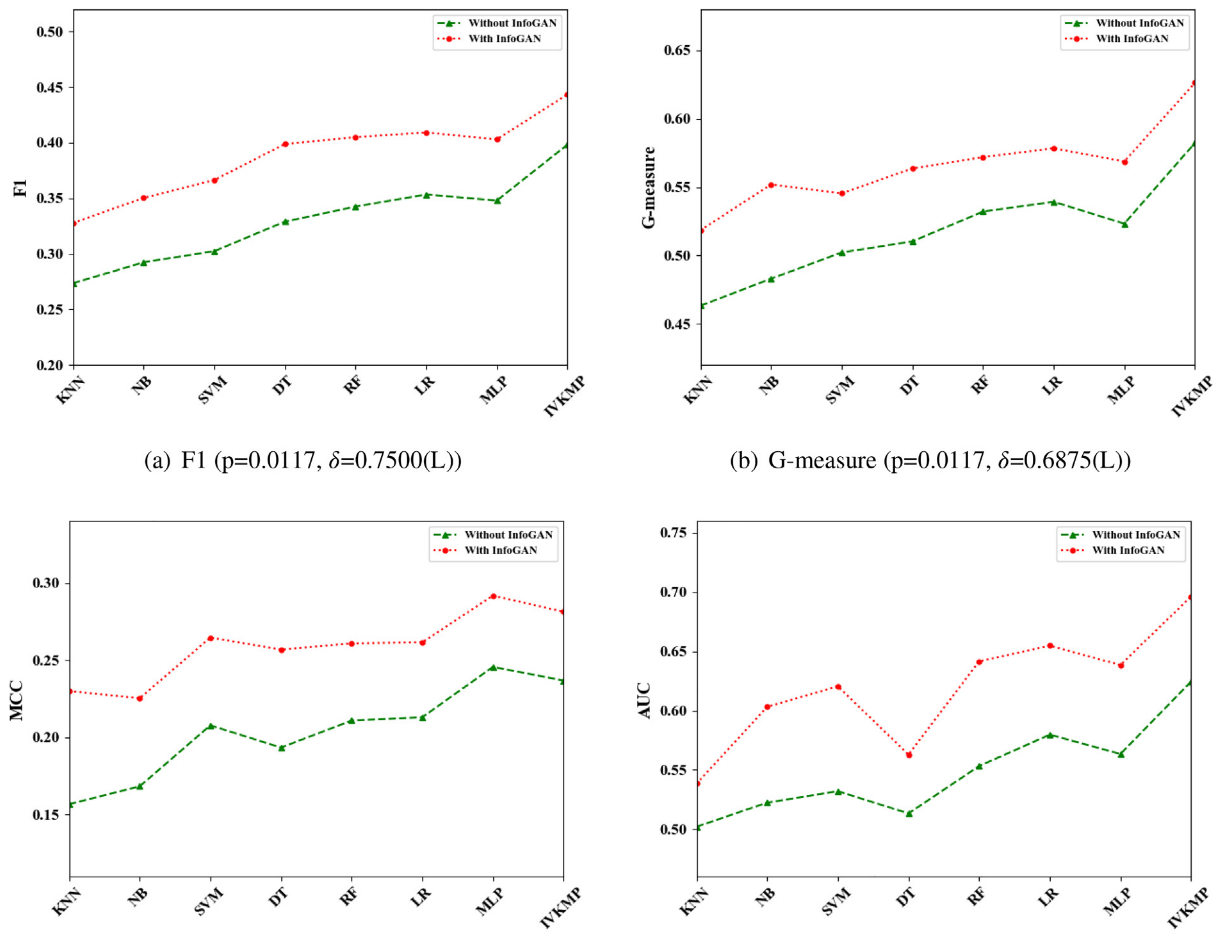
In this section, we first survey the influence of InfoGAN on our IVKMP model, and then report the three types of loss values and running time for generating defect instances on each source software project using the InfoGAN.

5.4.1. Influence for InfoGAN on the proposed IVKMP model

This experiment is designed to investigate whether or how much InfoGAN would affect our IVKMP model, so we compare IVKMP using InfoGAN with IVKMP without using InfoGAN across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC.

Table 13
P-value and Cliff's deltas (δ) for verifying generalization capacity of PCANet in IVKMP in terms of four indicators.

Against	F1		G-measure		MCC		AUC	
	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)
KNN	0.0077	0.9753(L)	0.0077	0.9753(L)	0.0077	0.8025(L)	0.0077	0.9753(L)
NB	0.0077	0.8765(L)	0.0077	0.9259(L)	0.0077	0.6790(L)	0.0077	0.9259(L)
SVM	0.0109	0.8519(L)	0.0077	0.8519(L)	0.2135	0.2099(S)	0.0209	0.7778(L)
DT	0.0209	0.3827(M)	0.0077	0.7531(L)	0.0663	0.4074(M)	0.0077	0.9753(L)
RF	0.0209	0.4568(M)	0.0109	0.5556(L)	0.0109	0.3086(S)	0.0109	0.6049(L)
LR	0.0108	0.3827(M)	0.0077	0.6296(L)	0.0506	0.2840(S)	0.0109	0.4321(M)
MLP	0.0284	0.3827(M)	0.0209	0.5062(L)	0.0858	−0.2099	0.0209	0.5062(L)



(a) F1 (p=0.0117, δ=0.7500(L))

(b) G-measure (p=0.0117, δ=0.6875(L))

(c) MCC (p=0.0117, δ=0.8750(L))

(d) AUC (p=0.0117, δ=0.7188(L))

Fig. 7. The HDP performance comparison for the IVKMP model using InfoGAN and without using InfoGAN in terms of four evaluation indicators.

Fig. 7 shows the HDP performance comparison for the IVKMP model using InfoGAN and without using InfoGAN across total heterogeneous project pairs of 26 target projects in terms of four evaluation indicators. The x-axis represents eight defect predictors; the y-axis represents different evaluation indicators. We count the average indicator values of eight multi-objective metric subset selection approaches on each defect predictor. Since our IVKMP model adopts the PCANet as the defect predictor, the performance of the PCANet predictor is expressed by IVKMP in Fig. 7. We can find that the IVKMP model with InfoGAN achieves the better HDP performance than the IVKMP model without InfoGAN in terms of four indicators. We employ the Wilcoxon signed-rank test and Cliff's delta (δ) to validate whether the HDP performance differences between IVKMP with InfoGAN and IVKMP without InfoGAN are statistically significant. In terms of four evaluation indicators, the p-values are less than 0.05 and the δ is larger than 0.474 (L), which indicates that the HDP performance differences between them are significant.

In conclusion, the InfoGAN can significantly enhance the prediction performance of the IVKMP model in terms of F1, G-measure, MCC and AUC.

5.4.2. Three types of loss values for InfoGAN on each source project

In this experiment, we explore the loss values for generating defect instances on each source software project using the deep generation model – InfoGAN, so as to more directly reflect the quality of the generated defect instances. As the first step of our IVKMP model, we employ InfoGAN to perform data augmentation for each of the original 26 software projects separately, and obtain its corresponding three types of loss values simultaneously based on each original software project. Table 14 shows three types of loss values for generating defect instances on each source software project using the deep generation model – InfoGAN, including discriminator loss, recognition loss, generator loss, in which the recognition loss refers to the loss of the recognition network used as an auxiliary distribution. The InfoGAN adopts the variational mutual information maximization to compute lower bounding mutual information by defining the auxiliary distribution (i.e., recognition network) $S(\vartheta|X)$ to approximate a posterior distribution $P(\vartheta|X)$, where X denotes the generated data and ϑ denotes the latent code. For example, for ant-1.3 in Table 14, InfoGAN is used to generate new defect instances based on the original ant-1.3, so as to obtain three types of loss values (i.e., Discriminator loss: 0.71; Recognition loss: 0.56; Generator loss: 0.01) of ant-1.3. Furthermore, from this table, we can also observe that the discriminator loss is large and the generator loss is small, which also demonstrates that the quality of the generated defect instances is quite high.

5.4.3. Running time for InfoGAN on each source project

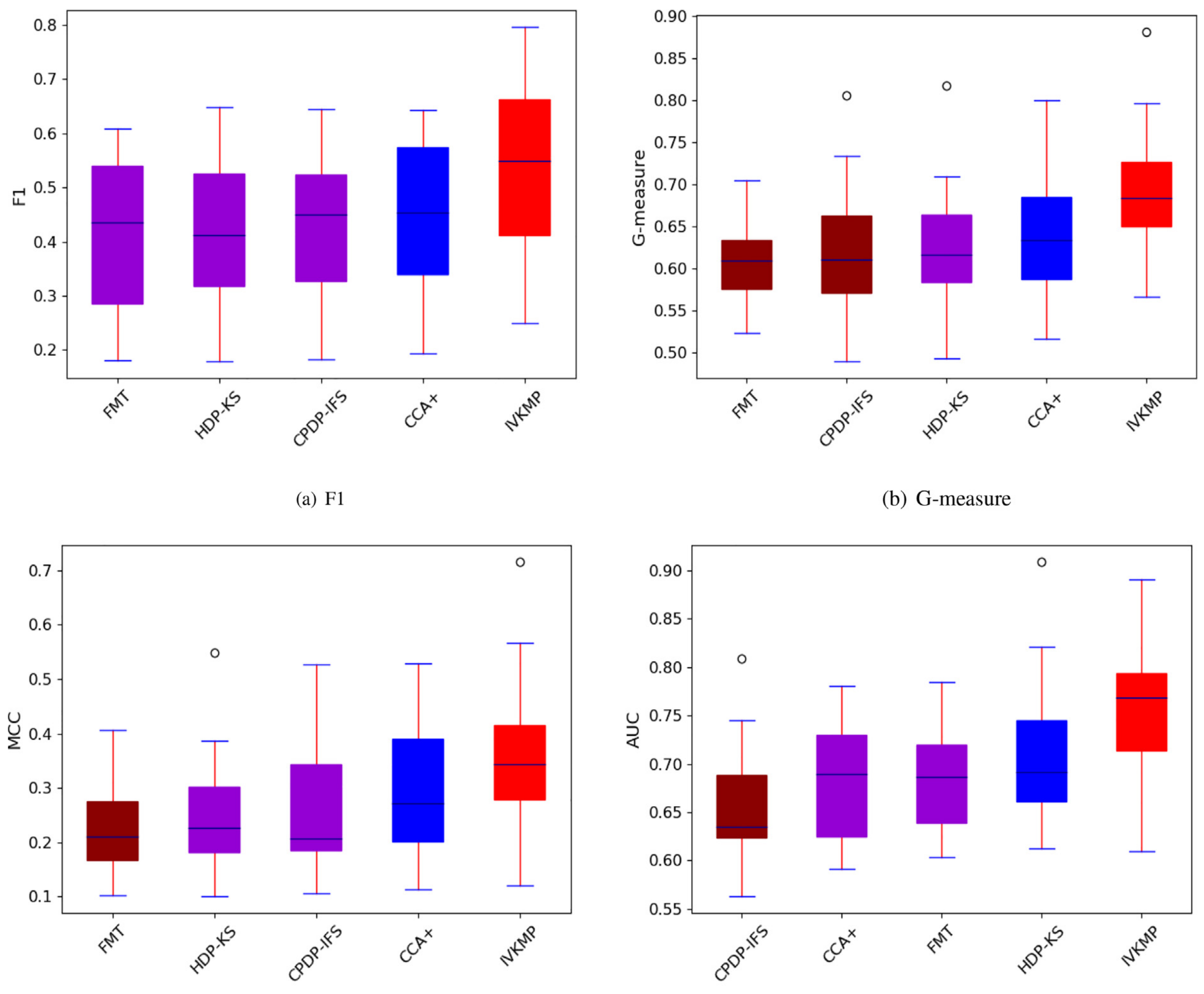
In this experiment, we report the running time for generating defect instances on 26 source software projects using the InfoGAN, as shown in Table 15. From this table, we can observe that the running time on different software projects ranges from 364.06 s for xalan-2.4 to 685.41 s for PC2. We believe that this running time is practical and feasible in actual software defect prediction.

Table 14
Three types of loss values for generating defect instances on each source software project using the InfoGAN.

Projects	Discriminator loss	Recognition loss	Generator loss	Projects	Discriminator loss	Recognition loss	Generator loss
ant-1.3	0.71	0.56	0.01	PC1	0.71	0.53	0
camel-1.0	0.83	0.67	0.04	PC2	0.78	0.49	0.01
poi-1.5	0.72	0.54	0	PC4	0.77	0.50	0.03
redaktor	0.71	0.56	0	JDT	0.83	0.78	0.31
xalan-2.4	0.72	0.55	0.02	LC	0.78	0.54	0.05
xerces-1.2	0.78	0.56	0.01	EQ	0.77	0.55	0.05
synapse-1.2	0.71	0.56	0.01	PDE	0.84	0.81	0.17
log4j-1.0	0.71	0.55	0.02	ML	0.89	0.82	0.19
Apache	0.77	0.54	0.01	AR1	0.71	0.55	0.01
Safe	0.71	0.56	0.02	AR3	0.80	0.77	0.24
ZXing	0.81	0.56	0.07	AR4	0.79	0.68	0.15
MW1	0.71	0.55	0	AR5	0.80	0.78	0.09
MC2	0.72	0.54	0.03	AR6	0.76	0.53	0.07

Table 15
Running time for generating defect instances on each source software project using the InfoGAN (in seconds).

Projects	Running time	Projects	Running time
ant-1.3	574.36	PC1	460.89
camel-1.0	373.97	PC2	685.41
poi-1.5	376.41	PC4	453.61
redaktor	368.81	JDT	488.44
xalan-2.4	364.06	LC	482.44
xerces-1.2	370.74	EQ	477.24
synapse-1.2	586.98	PDE	484.92
log4j-1.0	602.97	ML	487.65
Apache	421.81	ar1	419.27
Safe	423.30	ar3	655.66
ZXing	645.34	ar4	662.07
MW1	660.83	ar5	669.73
MC2	439.36	ar6	673.25



(a) F1 (b) G-measure

(c) MCC (d) AUC

Fig. 8. The Scott-Knott ESD ranking for IVKMP compared with four HDP models in terms of four indicators.

5.5. Superiority verification for the proposed IVKMP model

To more directly demonstrate the superiority of our IVKMP model, we compare it with four state-of-the-art HDP models in prediction performance across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC, including FMT, HDP-KS, CPDP-IFS, and CCA+.

To conduct a visual comparison of the HDP performance differences between the IVKMP and four state-of-the-art HDP models, we depict the boxplots with the Scott-Knott ESD test in terms of four evaluation indicators in Fig. 8. The blue line in each box denotes the median indicator value for each HDP model. The x-axis represents five HDP models; the y-axis denotes different evaluation indicators. From Fig. 8, we can observe that our IVKMP model is in the highest rank (the red

Table 16
P-value and Cliff's deltas (δ) for IVKMP compared with four HDP models in terms of four indicator.

Against	F1		G-measure		MCC		AUC	
	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)	p-value	δ(E)
FMT	1.67E−05	0.4290(M)	1.18E−05	0.6686(L)	1.05E−05	0.6183(L)	1.87E−05	0.6095(L)
HDP-KS	8.30E−06	0.4201(M)	2.63E−05	0.5547(L)	1.87E−05	0.5030(L)	5.68E−05	0.4660(M)
CPDP-IFS	1.67E−05	0.4142(M)	1.33E−05	0.5710(L)	1.18E−05	0.4527(L)	9.33E−06	0.7604(L)
CCA+	2.35E−05	0.3107(S)	0.0006	0.3905(M)	0.0002	0.2959(S)	1.87E−05	0.6287(L)

box) in terms of four evaluation indicators, which indicates that the IVKMP can enhance the HDP performance compared with four state-of-the-art HDP models. In addition, the median value (the blue line) achieved by IVKMP is higher than those achieved by four HDP models in terms of four evaluation indicators, which fully reflects the advantages of our IVKMP model.

We also employ the Wilcoxon signed-rank test and Cliff's delta (δ) to evaluate whether the performance differences between the IVKMP and four state-of-the-art HDP models are statistically significant. Table 16 shows p-values and Cliff's deltas (δ) of the IVKMP compared with four HDP models in terms of F1, G-measure, MCC and AUC. In terms of four evaluation indicators, the p-values are less than 0.05 (significant) and the δ is larger than 0.147 (not negligible), which is consistent with the observations in Fig. 8.

In summary, our IVKMP model can enhance the HDP performance compared with four state-of-the-art HDP models across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC.

6. Discussions

In this section, we discuss the possible reasons that affect the HDP performance of our data-driven IVKMP model corresponding to the previously designed experiments.

In Section 5.1, we explore whether our multi-objective VaEA optimization approach in IVKMP is superior to eight state-of-the-art multi-objective metric subset selection approaches in terms of the effect for HDP. We can conclude that the VaEA in IVKMP can achieve the optimal average HDP performance compared with eight state-of-the-art multi-objective metric subset selection approaches across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure, MCC and AUC. We analyze some possible reasons for this result. For the multi-objective VaEA optimization approach, the maximum-vector-angle-first principle used in the environmental selection can guarantee the wideness and uniformity of the solution sets (i.e., metric subsets), and worse solutions in terms of the convergence (evaluated by the sum of normalized objectives) can be conditionally replaced by other individuals according to the worse-elimination principle, so the search capability of the solutions can also be strengthened accordingly. Compared with VaEA, the other eight multi-objective metric subset selection approaches do not have these advantages. In addition, the multi-objective VaEA optimization approach can consider both convergence and diversity of the obtained solution sets (i.e., metric subsets) and well maintain a good balance between them. The multi-objective VaEA optimization approach can achieve the highest average HV value compared with the other eight multi-objective metric subset selection approaches, which also verifys that VaEA has the best convergence and diversity. Also, compared to the ordinary single-solution or single-objective metric subset selection approaches, these multi-objective metric subset selection approaches adopted in this paper can select multiple sets of metric subsets (multiple solutions) for source software projects, which can well compensate for the deficiency of this metric matching and transfer approach used that cannot search for suitable matching target metric and enhance the prediction performance of the HDP model to a certain extent. This also fully reflects the advantages of the multi-objective metric subset selection approaches. We also investigate the efficiency of our multi-objective VaEA optimization approach compared with eight state-of-the-art multi-objective metric subset selection approaches. The experimental results show that the VaEA takes the least average running time (171.07) on 26 source software projects. This may be because the VaEA is not affected by a set of weight vectors and has a better capacity to search for the pareto optimal solution sets (i.e., metric subsets), and the search speed is unable to slow down significantly. Moreover, the VaEA has less algorithmic parameters and its time complexity is relatively low.

In Section 5.2, we investigate whether the PCANet predictor in IVKMP performs better than seven classic defect predictors on HDP performance. We can draw the conclusion that the PCANet in IVKMP can enhance the HDP performance compared with seven classic defect predictors across total heterogeneous project pairs of 26 target projects in terms of F1, G-measure and AUC. Compared to ordinary machine learning algorithms, as a lightweight but effective deep learning network, the PCANet with the binary hashing and block-wise histograms can learn sufficient robustness and invariance in the final defect metrics, which is capable of essentially capturing more semantically related robust representations. Prior studies [49,16] have demonstrated that the abstract deep semantic metrics have stronger discriminating capacity for different classes (defective or non-defective). Chan et al. [1] have verified that the PCANet can achieve very competitive results in many classification tasks, such as texture classification, object recognition, which are already on par with, or often better than, state-of-the-art approaches. We also investigate the efficiency of the PCANet predictor in IVKMP compared with seven classic defect predictors. We can draw the conclusion that although the PCANet predictor takes the second longest running time (1.29s), we believe that this time cost is still acceptable, which is considered to be a lightweight deep defect predictor and is equivalent to sacrificing efficiency for effectiveness. The PCANet predictor consists of three processing stages. In the first two stages, we utilize the basic PCA filters to learn the two-stage convolution filter banks instead of the convolution operation in the traditional convolutional neural network. In the last output layer, we adopt the binary hashing for encoding and the nonlinear processing, and use block-wise histograms of the binary hashing encoding for pooling operations. The SVM classifier is used to predict whether each instance module is defective. Compared with six machine learning algorithms except MLP, not only the SVM classifier in the PCANet predictor needs training, but also the network structure of PCANet needs iterative training and takes a long time, so our PCANet predictor takes a long time. In addition, since MLP adopts gradient descent for training, it takes the longest running time compared to the other seven machine learning algorithms.

In Section 5.3, we demonstrate the generalization capabilities of the multi-objective VaEA optimization approach and PCANet predictor in IVKMP. We can conclude that our multi-objective VaEA optimization approach and PCANet predictor in IVKMP have strong robust generalization performance in terms of F1, G-measure, MCC and AUC compared with eight baseline multi-objective metric subset selection approaches and seven baseline defect predictors, which indicates that our multi-objective VaEA optimization approach and PCANet predictor can adapt to more complex data distribution or more kinds of datasets. The reasons for the predominant performance of VaEA and PCANet have been explained above. Furthermore, our IVKMP model employs the multi-objective VaEA optimization approach to select the representative defect metric subsets with strong robustness, which also enhances the generalization capacity of the PCANet predictor to a certain extent.

In Section 5.4, we investigate whether or how much InfoGAN would affect our IVKMP model. The experimental results show that the InfoGAN can significantly enhance the prediction performance of the IVKMP model in terms of F1, G-measure, MCC and AUC. Previous studies [50,62,45,15] have demonstrated that class imbalance can adversely affect or even seriously degrade the prediction performance of the model. The amount of training data is also an important factor affecting the prediction performance of the model. In this paper, we utilize the InfoGAN to perform data augmentation tasks, which not only achieves class balance, but also settles the issue of training data shortage for source software projects and provides sufficient training instances for the subsequent deep learning predictor – PCANet. Therefore, the InfoGAN can significantly boost the prediction performance of the IVKMP model.

In Section 5.5, we explore whether the proposed IVKMP model outperforms four state-of-the-art HDP models, including FMT, HDP-KS, CPDP-IFS, and CCA+. We can draw the conclusion that our IVKMP model can boost the HDP performance compared with four state-of-the-art HDP models in terms of F1, G-measure, MCC and AUC. We also analyze some possible reasons for this result. We employ the InfoGAN to achieve defect class balance and settle the issue of training data shortage for source software projects. In addition, we utilize multi-objective VaEA optimization approach to conduct metric selection for source software projects, so as to select the optimal representative metric subsets with strong robustness. Compared with the single-solution metric selection methods (such as chi-square, gain ratio, relief-F, significance attribute evaluation) in HDP-KS, the multi-objective VaEA optimization approach can select multiple sets of metric subsets (multiple solutions) for source software projects, which not only has the strong metric selection capacity, but also well compensates for the deficiency of this metric matching and transfer approach (i.e., KSTest, maximum weighted bipartite matching) used that cannot search for suitable matching target metric and enhance the prediction performance of the HDP model to a certain extent. This also fully reflects the advantages of the multi-objective VaEA metric subset selection approach. Also, compared to the traditional machine learning classifiers (such as LR, NB, SVM) used in these four HDP models, the built deep defect predictor PCANet can essentially capturing more semantically related robust representations, and these abstract deep semantic features have stronger discriminating capacity for different classes (defective or non-defective). These reasons all contribute to the performance of our IVKMP model over four state-of-the-art HDP models.

7. Conclusion and future work

Heterogeneous defect prediction (HDP) is very promising and practical since it can exploit potentially heterogeneous data with different defect metrics for defect prediction, which provides a new practical guide for software developers or maintainers to conduct defect prediction across different software projects. In this study, we present a novel data-driven HDP model called IVKMP based on deep learning techniques and multi-objective optimization. The model first leverages an advanced deep generation model InfoGAN to achieve defect class balance and provide sufficient training instances for the subsequent deep learning model. Then the model employs the multi-objective VaEA optimization to select the fewest representative metric subsets while achieving the minimum error for source software projects. After completing metric matching and transfer using KS-Test and maximum weighted bipartite matching, we build a powerful defect predictor based on the lightweight but effective deep learning network PCANet with SVM, which is capable of essentially capturing semantically related robust representations. To verify the performance of our IVKMP model, we conduct extensive and large-scale experiments compared with multiple state-of-the-art baseline models across 542 heterogeneous project pairs of 26 software projects. The experimental results exhibit that our IVKMP model can promote the HDP performance.

In future work, we will explore new heterogeneous metric matching approaches, such as heterogeneous information network embedding technique, to achieve the matching of heterogeneous defect metrics across different software projects. Moreover, we plan to validate our IVKMP model in more open source software projects.

CRediT authorship contribution statement

Kun Zhu: Methodology, Validation, Writing – original draft. **Shi Ying:** Supervision, Methodology. **Weiping Ding:** Methodology, Writing – review & editing. **Nana Zhang:** Methodology, Validation. **Dandan Zhu:** Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to express their sincere appreciation to the anonymous reviewers for their insightful comments, which greatly improved the quality of this paper. This work is supported in part by the National Science Foundation of China (No. 62072342, NO. 61672392, NO. 61976120), and in part by the National Key Research and Development Program of China (No. 2016YFC1202204), and in part by the Natural Science Foundation of Jiangsu Province (NO. BK20191445), and in part by the Natural Science Key Foundation of Jiangsu Education Department under Grant 21KJA510004, and sponsored by Qing Lan Project of Jiangsu Province.

References

[1] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, Pcanet: A simple deep learning baseline for image classification?, IEEE Trans Image Process. 24 (2015) 5017–5032.
[2] B. Chen, W. Zeng, Y. Lin, D. Zhang, A new local search-based multiobjective optimization algorithm, IEEE Trans. Evol. Comput. 19 (2015) 50–73.
[3] H. Chen, Y. Tian, W. Pedrycz, G. Wu, R. Wang, L. Wang, Hyperplane assisted evolutionary algorithm for many-objective optimization problems, IEEE Trans. Cybern. 50 (2020) 3367–3380.
[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5–10, 2016, Barcelona, Spain, 2016, pp. 2172–2180..
[5] X. Chen, Y. Mu, K. Liu, Z. Cui, C. Ni, Revisiting heterogeneous defect prediction methods: How far are we?, Inf Softw. Technol. 130 (2021) 106441.
[6] M. Cheng, G. Wu, M. Jiang, H. Wan, G. You, M. Yuan, Heterogeneous defect prediction via exploiting correlation subspace, in: The 28th International Conference on Software Engineering and Knowledge Engineering, SEKE 2016, Redwood City, San Francisco Bay, USA, July 1–3, 2016, 2016, pp. 171–176..
[7] A. Ciurumelea, S. Proksch, H.C. Gall, Suggesting comment completions for python using neural language models, in: 27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18–21, 2020, 2020, pp. 456–467..
[8] G.W. Corder, D.I. Foreman, Nonparametric statistics for nonstatisticians: A step-by-step approach 87 (2009) 206–220.
[9] M. D'Ambros, M. Lanza, R. Robbes, Evaluating defect prediction approaches: a benchmark and an extensive comparison, Empir. Softw. Eng. 17 (2012) 531–577.
[10] G. Fan, X. Diao, H. Yu, K. Yang, L. Chen, Deep semantic feature learning with embedded static metrics for software defect prediction, in: 26th Asia-Pacific Software Engineering Conference, APSEC 2019, Putrajaya, Malaysia, December 2–5, 2019, 2019, pp. 244–251..
[11] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, S. Gnesi, Detecting requirements defects with NLP patterns: an industrial experience in the railway domain, Empir. Softw. Eng. 23 (2018) 3684–3733.
[12] J. Frank, J. Massey, The kolmogorov-smirnov test for goodness of fit, Publ. Am. Statal Assoc. 46 (1951) 68–78.
[13] B. Gärtner, J. Matousek, Understanding and using linear programming, Universitext, Springer, 2007.
[14] S.M. Ghaffarian, H.R. Shahriari, Neural software vulnerability analysis using rich intermediate graph representations of programs, Inf. Sci. 553 (2021) 189–207.
[15] L. Goel, M. Sharma, S.K. Khatri, D. Damodaran, Cross-project defect prediction using data sampling for class imbalance learning: an empirical study, Int. J. Parallel Emergent Distrib. Syst. (2019) 1–14.
[16] J. Guo, J. Cheng, J. Cleland-Huang, Semantically enhanced software traceability using deep learning techniques, in: Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20–28, 2017, IEEE/ ACM, 2017, pp. 3–14..
[17] J. Guo, Y. Yuan, C. Zhang, Joint supervision for discriminative feature learning in convolutional neural networks, in: Computer Vision – Second CCF Chinese Conference, CCCV 2017, Tianjin, China, October 11–14, 2017, Proceedings, Part II, 2017, pp. 509–520..
[18] H. Ha, H. Zhang, Deepperf: performance prediction for configurable software with deep sparse neural network, in: Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25–31, 2019, 2019, pp. 1095–1106..
[19] M.A. Hall, L.A. Smith, Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper, in: Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference, May 1-5, 1999, Orlando, Florida, USA, 1999, pp. 235–239..
[20] P. He, B. Li, Y. Ma, Towards cross-project defect prediction with imbalanced feature sets, CoRR (2014), abs/1411.4228.
[21] Z. He, F. Shu, Y. Yang, M. Li, Q. Wang, An investigation on the feasibility of cross-project defect prediction, Autom. Softw. Eng. 19 (2012) 167–199.
[22] Z. He, G.G. Yen, Many-objective evolutionary algorithm: Objective space reduction and diversity improvement, IEEE Trans. Evol. Comput. 20 (2016) 145–160.
[23] C. Hsu, C. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Networks 13 (2002) 415–425.
[24] X. Huo, M. Li, Z. Zhou, Learning unified features from natural and programming languages for locating buggy source code, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, 2016, pp. 1606–1612..
[25] C. Igel, N. Hansen, S. Roth, Covariance matrix adaptation for multi-objective optimization, Evol. Comput. 15 (2007) 1–28.
[26] X. Jing, F. Wu, X. Dong, F. Qi, B. Xu, Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 – September 4, 2015, 2015, pp. 496–507..
[27] M. Jureczko, L. Madeyski, Towards identifying software project clusters with regard to defect prediction, in, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE 2010, September 12–13, 2010, Timisoara, Romania, 2010, p. 9.
[28] M. Li, S. Yang, X. Liu, Pareto or non-pareto: Bi-criterion evolution in multiobjective optimization, IEEE Trans. Evol. Comput. 20 (2016) 645–665.
[29] Z. Li, X. Jing, X. Zhu, H. Zhang, B. Xu, S. Ying, Heterogeneous defect prediction with two-stage ensemble learning, Autom. Softw. Eng. 26 (2019) 599–651.
[30] Z. Li, X. Jing, X. Zhu, H. Zhang, B. Xu, S. Ying, On the multiple sources and privacy preservation issues for heterogeneous defect prediction, IEEE Trans. Software Eng. 45 (2019) 391–411.
[31] H. Liu, L. Chen, Q. Zhang, K. Deb, An evolutionary many-objective optimisation algorithm with adaptive region decomposition, in: IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24–29, 2016, 2016, pp. 4763–4769..
[32] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, Y. Wang, Deepmutation: Mutation testing of deep learning systems, in: 29th IEEE International Symposium on Software Reliability Engineering, ISSRE 2018, Memphis, TN, USA, October 15–18, 2018, 2018, pp. 100–111..
[33] M.M. Mafarja, S. Mirjalili, Hybrid whale optimization algorithm with simulated annealing for feature selection, Neurocomputing 260 (2017) 302–312.
[34] D. Miholca, G. Czibula, I.G. Czibula, A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks, Inf. Sci. 441 (2018) 152–170.
[35] J. Nam, W. Fu, S. Kim, T. Menzies, L. Tan, Heterogeneous defect prediction, IEEE Trans. Software Eng. 44 (2018) 874–896.
[36] D.N. Palacio, D. McCrystal, K. Moran, C. Bernal-Cárdenas, D. Poshyvanyk, C. Shenefiel, Learning to identify security-related issues using convolutional neural networks, in: 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019, Cleveland, OH, USA, September 29 - October 4, 2019, 2019, pp. 140–144..

[37] A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: An astounding baseline for recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23–28, 2014, 2014, pp. 512–519..
[38] M.J. Shepperd, Q. Song, Z. Sun, C. Mair, Data quality: Some comments on the NASA software defect datasets, IEEE Trans. Software Eng. 39 (2013) 1208–1215.
[39] C. Shi, B. Hu, W.X. Zhao, P.S. Yu, Heterogeneous information network embedding for recommendation, IEEE Trans. Knowl. Data Eng. 31 (2019) 357–370.
[40] C. Shi, Y. Li, J. Zhang, Y. Sun, P.S. Yu, A survey of heterogeneous information network analysis, IEEE Trans. Knowl. Data Eng. 29 (2017) 17–37.
[41] Y. Sun, J. Han, Mining Heterogeneous Information Networks: Principles and Methodologies. Synthesis Lectures on Data Mining and Knowledge Discovery, Morgan & Claypool Publishers, 2012.
[42] Z. Sun, J. Li, H. Sun, L. He, CFPS: collaborative filtering based source projects selection for cross-project defect prediction, Appl. Soft Comput. 99 (2021) 106940.
[43] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, K. Matsumoto, An empirical comparison of model validation techniques for defect prediction models, IEEE Trans. Software Eng. 43 (2017) 1–18.
[44] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, K. Matsumoto, The impact of automated parameter optimization on defect prediction models, IEEE Trans. Software Eng. 45 (2019) 683–711.
[45] H. Tong, B. Liu, S. Wang, Q. Li, Transfer-learning oriented class imbalance learning for cross-project defect prediction, CoRR (2019), abs/1901.08429.
[46] B. Turhan, T. Menzies, A.B. Bener, J.S.D. Stefano, On the relative value of cross-company and within-company data for defect prediction, Empir. Softw. Eng. 14 (2009) 540–578.
[47] K. Wang, L. Liu, C. Yuan, Z. Wang, Software defect prediction model based on LASSO-SVM, Neural Comput. Appl. 33 (2021) 8249–8259.
[48] R. Wang, R.C. Purshouse, P.J. Fleming, Preference-inspired coevolutionary algorithms for many-objective optimization, IEEE Trans. Evol. Comput. 17 (2013) 474–494.
[49] S. Wang, T. Liu, L. Tan, Automatically learning semantic features for defect prediction, in: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14–22, 2016, ACM, 2016, pp. 297–308..
[50] S. Wang, X. Yao, Using class imbalance learning for software defect prediction, IEEE Trans. Reliab. 62 (2013) 434–443.
[51] W. Wang, G. Li, B. Ma, X. Xia, Z. Jin, Detecting code clones with graph neural network and flow-augmented abstract syntax tree, in: 27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18–21, 2020, 2020, pp. 261–271..
[52] H. Wei, C. Hu, S. Chen, Y. Xue, Q. Zhang, Establishing a software defect prediction model via effective dimension reduction, Inf. Sci. 477 (2019) 399–409.
[53] Y. Wei, X. Wang, W. Guan, L. Nie, Z. Lin, B. Chen, Neural multimodal cooperative learning toward micro-video understanding, IEEE Trans. Image Process. 29 (2020) 1–14.
[54] J. van de Wolfshaar, M.F. Karaaba, M.A. Wiering, Deep convolutional neural networks and support vector machines for gender recognition, in: IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7–10, 2015, 2015, pp. 188–195..
[55] R. Wu, H. Zhang, S. Kim, S. Cheung, Relink: recovering links between bugs and changes, in: SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5–9, 2011, 2011, pp. 15–25..
[56] X. Xia, D. Lo, S.J. Pan, N. Nagappan, X. Wang, HYDRA: massively compositional model for cross-project defect prediction, IEEE Trans. Software Eng. 42 (2016) 977–998.
[57] Y. Xiang, Y. Zhou, M. Li, Z. Chen, A vector angle-based evolutionary algorithm for unconstrained many-objective optimization, IEEE Trans. Evol. Comput. 21 (2017) 131–152.
[58] X. Xu, J. Chen, H. Zhang, W.W.Y. Ng, D4net: De-deformation defect detection network for non-rigid products with large patterns, Inf. Sci. 547 (2021) 763–776.
[59] Z. Xu, J. Liu, Z. Yang, G. An, X. Jia, The impact of feature selection on defect prediction performance: An empirical comparison, in: 27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23–27, 2016, 2016, pp. 309–320..
[60] X. Yang, D. Lo, X. Xia, J. Sun, TLEL: A two-layer ensemble learning approach for just-in-time defect prediction, Inf. Softw. Technol. 87 (2017) 206–220.
[61] Q. Yu, S. Jiang, Y. Zhang, A feature matching and transfer approach for cross-company defect prediction, J. Syst. Softw. 132 (2017) 366–378.
[62] Yu, X., Zhou, M., Chen, X., Deng, L., 2017b. Using class imbalance learning for cross-company defect prediction, in: The 29th International Conference on Software Engineering and Knowledge Engineering, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 5–7, 2017, pp. 117–122..
[63] Y. Yuan, H. Xu, B. Wang, B. Zhang, X. Yao, Balancing convergence and diversity in decomposition-based many-objective optimizers, IEEE Trans. Evol. Comput. 20 (2016) 180–198.
[64] N. Zhang, S. Ying, W. Ding, K. Zhu, D. Zhu, WGNCS: A robust hybrid cross-version defect model via multi-objective optimization and deep enhanced feature representation, Inf. Sci. 570 (2021) 545–576.
[65] N. Zhang, K. Zhu, S. Ying, X. Wang, Software defect prediction based on stacked contractive autoencoder and multi-objective optimization, CMC-Comput. Mater. Continua 65 (2020) 279–308.
[66] Q. Zhang, A. Zhou, Y. Jin, RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm, IEEE Trans. Evol. Comput. 12 (2008) 41–63.
[67] X. Zhang, Y. Tian, Y. Jin, A knee point-driven evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 19 (2015) 761–776.
[68] T. Zhou, X. Sun, X. Xia, B. Li, X. Chen, Improving defect prediction with deep forest, Inf. Softw. Technol. 114 (2019) 204–216.
[69] K. Zhu, S. Ying, N. Zhang, D. Zhu, Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network, J. Syst. Softw. 180 (2021) 111026.
[70] K. Zhu, N. Zhang, S. Ying, D. Zhu, Within-project and cross-project just-in-time defect prediction based on denoising autoencoder and convolutional neural network, IET Softw. 14 (2020) 185–195.
[71] K. Zhu, N. Zhang, Q. Zhang, S. Ying, X. Wang, Software defect prediction based on non-linear manifold learning and hybrid deep learning techniques, CMC-Comput. Mater. Continua 65 (2020) 1467–1486.