Correlation feature and instance weights transfer learning for cross project software defect prediction

Quanyi Zou1 | Lu Lu2,3 | Shaojian Qiu4 | Xiaowei Gu1 | Ziyi Cai2

1The School of Software Engineering, South China University of Technology, Guangzhou, China
2The School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
3Modern Industrial Technology Research Institute, South China University of Technology, Zhongshan, China
4The College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China

Abstract
Due to the differentiation between training and testing data in the feature space, cross-project defect prediction (CPDP) remains unaddressed within the field of traditional machine learning. Recently, transfer learning has become a research hot-spot for building classifiers in the target domain using the data from the related source domains. To implement better CPDP models, recent studies focus on either feature transferring or instance transferring to weaken the impact of irrelevant cross-project data. Instead, this work proposes a dual weighting mechanism to aid the learning process, considering both feature transferring and instance transferring. In our method, a local data gravitation between source and target domains determines instance weight, while features that are highly correlated with the learning task, uncorrelated with other features and minimizing the difference between the domains are rewarded with a higher feature weight. Experiments on 25 real-world datasets indicate that the proposed approach outperforms the existing CPDP methods in most cases. By assigning weights based on the different contribution of features and instances to the predictor, the proposed approach is able to build a better CPDP model and demonstrates substantial improvements over the state-of-the-art CPDP models.

1 | INTRODUCTION

Software Defect Prediction (SDP) can conveniently assist the quality assurance team to reasonably prioritize and allocate testing resources by identifying the modules or classes that are potentially defective [1]. A majority of existing defect prediction methodologies leverage machine learning and data mining approaches to build a predictor [2–5]. These approaches can successfully identify defective modules or classes when training data and test data are from the same project. Thus, they are called the Within Project Defect Prediction (WPDP) model. However, in practice, most software companies or organizations seldom collect data to train the defect predictors, either because defective modules and classes labelling is too expensive, or because there are not sufficient training data at an early phase. Fortunately, there exist sufficient labelled data from public data repositories provided by different software companies or organizations. Many researchers have presented several approaches to study SDP using data from these public repositories. Different from WPDP, the source and target projects in Cross-Project Defect Prediction (CPDP) scenarios are not necessarily from the same project. Traditional machine learning methods may not work well in CPDP scenarios as the source and target projects may be entirely different at feature and instance level. To address this issue, transfer learning provides feasible techniques. The related auxiliary information from the source project can be used for the target project by implementing transfer learning [6,7]. The specific methods of transfer learning can be broadly categorized into instance-transfer learning and feature-transfer learning [8,9].

Instance-transfer learning can be subdivided into instance-filter transfer and instance-weighting transfer. In instance-filter transfer, corresponding instances related with the target are selected from the source project to train a CPDP model. Turhan et al. [10] propose the Nearest Neighbour filter (NN filter) model, which filters instances based on the nearest neighbour method. Kazuya Kawata et al. [11] and Peters et al.
Feature-transfer learning can be subdivided into feature-representation transfer and feature-selection transfer. In feature-representation transfer learning, two projects are transferred to a common high-dimensional feature representation space where the distance of distribution between the two projects is minimized. Transfer Component Analysis plus (TCA+) \cite{14,15} is the state-of-the-art feature-representation transfer learning approach for CPDP. In feature-selecting transfer learning, appropriate features are selected to train a CPDP model. Yu et al. \cite{16} proposed a feature-selecting transfer learning approach, which evaluates the individual predictive ability of each feature and the redundancy between different features. Menzies et al. \cite{17} reported the sub-features Naive Bayes (sub + NB) for SDP and obtained the best results by using log filtering to select sub-features.

Instance-filter and feature-selection transfer learning focus more on filtering the source project’s information to weaken the impact of irrelevant cross-project data, which results in wasting part of the data information. Instead, this work mainly focuses on the importance of instances and features to the predictive model, and expresses their importance in terms of weight. Appropriate feature and instance weights are more flexible than feature selection (FS) and instance filtering. For example, setting the weights of some features or instances to 1.0 and the others to 0.0 is equivalent to FS and instance filtering. Feature and instance weighting methods are able to assign continuous positive weights, while FS and instance filter methods merely specify discrete values. Therefore, the authors propose a novel CPDP model named Correlation Feature and Instance Weighting Transfer Naive Bayes (CFIW-TNB) by weighting for each feature and instance rather than single instance filter or FS. Despite its simple structure, Naive Bayes Classifier (NBC) has continuously been rated as one of the top 10 classic algorithms \cite{18}. Thus, using NBC to predict software defect is also advocated by some scholars. The citing predictive performance and comprehensibility are taken as its major advantages \cite{19,20}. NBC’s approaches are summarized into six main categories by Jiang et al. \cite{21}: (1) structure extension; (2) feature selection; (3) feature weighting; (4) instance selection; (5) instance weighting; and (6) fine Tuning. Of numerous NBCs to alleviating conditional independence assumption, feature weighting (instance weighting) has placed the focus more on highly predictive features (instances) than those that are less predictive. Some approaches proposed in Ref. \cite{23–26} belong to the feature weighting NBC model to predict software defects. Intuitively, the highly predictive features should be assigned bigger weights when training a prediction model. Both TNB by proposed Ma et al. \cite{13} and a Credibility theory-based Naive Bayes (CNB) proposed by Poon et al. \cite{27} is the instance weighting NBC model for CPDP. The approaches do not directly filter instances, but mitigate the impact of irrelevant data by reducing the weight of the instance. Our proposed CFIW-TNB is an NBC that combines instance weighting NBC and feature weighting NBC.

We validate the proposed CFIW-TNB against existing five state-of-the-art CPDP approaches and single weighting NBC (TNB, CF-NB) \cite{21} using 25 real-world datasets from the PROMISE data repository. AUC, F-measure, pd and G-measure are adopted as performance measurement. To statistically analyze the performance between our method and other baseline methods, we perform the non-parametric Frideman test with the Nemenyi’s post-hoc test. The authors address the following research questions (RQ):

- **RQ1** How effective is CFIW-TNB method compared with other CPDP methods?
- **RQ2** Is dual weighting mechanism helpful for CPDP?
- **RQ3** Is CFIW-TNB superior to its variants?

The main contributions of this work are:

1. To better reflect the importance of feature and instance, we proposed a novel CPDP model called CFIW-TNB, which considers both instance-transfer learning and feature-transfer learning rather than a single transferring strategy. It effectively improves the performance of single transfer learning methods.
2. CFIW-TNB enhances the local data gravitation method via considering the density of data distribution. The denser the data distribution, the greater the gravitation, and vice versa. Therefore, it is more flexible in allocating the instance weights than TNB.
3. We validate the effectiveness of our proposed methods, CFIW-TNB, by conducting extensive experiments on 25 real-world datasets.

The remainder of the article is organized as follows: Section 2 reviews relevant works. Section 3 proposes our method. Section 4 introduces the experimental setup. Section 5 shows the experimental results and analysis. Section 6 explains threats to validity. Section 7 concludes this work and proposes future work.

## 2 RELATED WORK

Within this section, we introduce related works on the feature and instance weighting NBC, and Naive Bayes for SDP.

### 2.1 Feature weighting NBC

A feature weighting NBC places more emphasis on the fact that features have differently predictive power, which assigns features weight based on their contribution to predictor. They are commonly called Feature Weighting Naive Bayes (FWNB). The formula for FWNB embedding feature weights is given as:
where $w_i \in R^+$ is the weight of the $i$th feature $A_i$, $a_i$ is the value of $i$th feature $A_i$, $P(c)$ is the prior probability, and $P(a_i | c)$ is the conditional probability. $P(c)$ and $P(a_i | c)$ can be obtained using Equations (2) and (3), respectively:

$$p(c) = \frac{1 + \sum_{j=1}^{n} \delta(c_j, c)}{q + n},$$

$$p(a_i | c) = \frac{1 + \sum_{j=1}^{n} \delta(a_{ij}, a_i) \delta(c_j, c)}{\sum_{i} \delta(c_j, c) + n_i},$$

where $n$ is the number of training instances, $n_i$ is the number of values for the $i$th feature, $q$ is the number of classes, $c_j$ is the class label of the $j$th training instance, $a_{ij}$ is the $i$th feature value of the $j$th training instance, $x = [a_{11}, a_{12}, \ldots, a_{mn}]$, and $\delta(\cdot)$ is the indicator function, which is one if its two parameters are identical and zero otherwise. How to define the weight of each feature is crucial and has attracted attention from more researchers. A FWNB can be briefly depicted as Algorithm 1.

### Algorithm 1 FWNB algorithm

**Input:** training dataset: $D$, an un-label test instance: $x$.
**Output:** test instance prediction label for test cases: $c(x)$.

1. Compute weights of features by a certain method;
2. Train an NBC based on $D$ and weights of features;
3. Predict a label $c(x)$ of $x$ by using the NBC;
4. Return $c(x)$.

### 2.2 Instance weighting NBC

An instance weighting NBC places emphasis on assigning weights for each instance in the training stage to reflect their predictive capability. These approaches are commonly called Instance Weighting Naive Bayes (IWNB). IWNB incorporates instance weights into the formula to give:

$$c(x) = \text{argmax} \ P(c) \prod_{i=1}^{m} P(a_i | c),$$

where $P(c)$ and $P(a_i | c)$ can be computed using Equations (5) and (6), respectively:

$$p(c) = \frac{1 + \sum_{j=1}^{n} w_j \delta(c_j, c)}{q + n},$$

$$p(a_i | c) = \frac{1 + \sum_{j=1}^{n} w_j \delta(a_{ij}, a_i) \delta(c_j, c)}{\sum_{i} w_j \delta(c_j, c) + n_i},$$

where $w_j$ is the weight of the $j$th instance. An IWNB can be briefly depicted as Algorithm 2.

### Algorithm 2 IWNB algorithm

**Input:** training dataset: $D$, an un-label test instance: $x$.
**Output:** test instance prediction label for test cases: $c(x)$.

1. Compute weights of instance by a certain method;
2. Train an NBC based on $D$ and weights of features;
3. Predict a label $c(x)$ of $x$ by using the NBC;
4. Return $c(x)$.

### 2.3 Transfer NBC

TNB proposed by Ma et al. [13] is instance weighting NBC. Its crucial problem is reweighting for instances in the training process. An instance can be expressed as $x = [a_{11}, a_{12}, \ldots, a_{mn}]$. $a_{ij}$ is the $i$th feature value for $x_i$, $m$ is the number of features. In TNB, the maximum and minimum values of each feature on the test set firstly are found. Then two vectors $\text{Max} = [Max_1, Max_2, \ldots, Max_m]$ and $\text{Min} = [Min_1, Min_2, \ldots, Min_m]$ are obtained. For instance $x_j$ from training set, degrees of similarity $s_j$ can be defined in:

$$s_j = \sum_{i=1}^{n} b(a_{ij}),$$

where $b(a_{ij}) = \begin{cases} 1 & \text{if Min} \leq a_{ij} \leq \text{Max} \\ 0 & \text{otherwise} \end{cases}$

TNB introduces the data gravitation so as to transfer the test set information into the classifier. Newton's universal gravitation law provides a unified description of gravitation between any two objects as a geometric property of space time [22]. The gravitation is proportional to the product of the two masses, while inversely proportional to the square of the distance between these two objects:

$$F = G \frac{m_1 m_2}{r^2},$$

where $G$ is the gravitational constant, $m_1$ and $m_2$, respectively, are the mass of two objects, $r$ is the distance between these two objects and $F$ is the gravitation between the two objects. TNB applies the universal gravitation to simulate data gravitation between training set and test set, and data gravitation is employed as the weight of instance. TNB supposes that one feature has mass $M$, and the mass of the test set is $mM$ and the mass of each training data is $s_j M$. Thus, the weight of instance $x_j$ is proportional to $m s_j M^2$ and inversely proportional to $r^2 = (m - s_j + 1)^2$. Therefore, the weight of instance can be written as follows [13]:

$$w_j = F_j = \frac{s_j}{(m - s_j + 1)^2},$$
If all the feature values locate between the maximum and minimum value for a certain instance, the \( r^2 = (m - s_j + 1)^2 \) is equal to 1. This moment, this instance is assigned the greatest weight. TNB classifier is briefly depicted as Algorithm 3.

### Algorithm 3 TNB algorithm

**Input:** training dataset: \( D \), an un-label test dataset: \( L \).

**Output:** test instance prediction label for test cases: \( c(\mathbf{x}) \).

1. Compute Max and Min vectors;
2. for each training instance \( x_i \in D \) do.
   1. Compute \( s_j \) of each instance \( x_i \) by Equation (7);
   2. Compute \( w_j \) of each instance \( x_i \) by Equation (9);
3. end for.
4. Train an NBC based on \( D \) and weights of instance;
5. Predict label \( c(\mathbf{x}) \) of \( \mathbf{x} \in L \) by using NBC;
6. Return \( c(\mathbf{x}) \).

### 2.4 NBC for software defect prediction

Malhotra et al. [28] have given a statistics that NBC accounts for 47.7% among all the SDP studies. In 2019, Hosseini et al. [29] have reviewed previous studies, and then concluded that Naive Bayes is the most commonly used modelling approach in SDP. These surveys reveal that there exists a room for research to improve NBC for SDP by alleviating the all feature fully independence assumption. For FS NBC, removing redundant and irrelevant features from raw features reduces the dimensionality of the data and improves the effectiveness of the learning algorithm. Menzies et al. [17] reported the sub-features Naive Bayes (sub + NB), belonging to the FS NBC model, and obtained the best results by using log filtering to select sub-features. Hosseini et al. [30] adopted information gain for FS. He et al. [31] used the FS technique to identify and remove irrelevant features to improve the effectiveness of predictors for SDP.

For feature-weighting NBC, the highly predictive feature deserves a larger weight, and vice versa. Turhan et al. [23,24] presented a series of feature weighting NBC models to alleviate the independence assumption by the feature weighting technique. Besides, Asmono et al. [25] proposed a feature weighting NBC to predict software defects by employing the absolute value of the correlation coefficient as the weighting technique. Recently, Huang et al. [26] utilized information diffusion as weights of feature to learn NBC.

Many studies have proposed some effective NBC for CPDP. For example, in Ref. [32], a universal defect predictor applying NBC as the modelling technique provides the context-aware rank transformation to overcome the difference in distribution between the source and target projects. In TNB proposed by Ma et al. [13], instances in source project are weighted according to data gravitation of instances between source and target projects. CNB [27] employs credibility theory to define instance weight. TNB and CNB belong into instance weighting NBC. Ryu et al. [33] proposed a multi-objective naive Bayes (MONB) considering the class imbalance problem for CPDP.

In addition, there exists structure extension NBC increasing the dependence between features to predict software defect. For example, Arar et al. [34] proposed the feature dependent naive Bayes (FDBN) to alleviate the independent assumption between features.

### 3 PROPOSED METHOD

In this section, we present the proposed CFIW-TNB model in detail. At the end of this section, we have given the whole pseudo-code of the CFIW-TNB model.

#### 3.1 Instance weight

In TNB, applying \( h(a_{ji}) \) determines degrees of similarity \( s_j \) between instances in training set and test set. The value of \( h(a_{ji}) \) is equal to 1, when \( a_{ji} \) is any value between the maximum value and the minimum value, which indicates that \( a_{ji} \) is evenly scattered between the maximum and minimum values. In other words, instances from test set and training set are objects with uniform mass. However, in fact, \( a_{ji} \) is not evenly scattered between the maximum and the minimum. For example, four features (amc, locm, locm3, ic) from the ant-1.4 software project are shown in Figure 1. They are not evenly spread between the maximum and the minimum. As shown in Figure 1, the distribution is denser between 0 and 40 than between 100 and 200. Thus, the gravitation generated by different parts also varies.

The denser the data distribution, the greater the gravitation, and the sparse the data distribution, the less the gravitation. Therefore, we introduce local data gravitation to ameliorate the data gravitation scheme. For instance \( x_i \) in the training set, degrees of similarity \( s_j^i \) between it and the test set can be rewritten as follows:

\[
s_j^i = \sum_{i=1}^{m} h^*(a_{ji}),
\]

where

\[
h^*(a_{ji}) = \begin{cases} 
\frac{1}{1 + \sum_{i=1}^{m} |a_{ji} - a_{ji}|} & \text{if } \text{Max}_{i} \leq a_{ji} \leq \text{Min}_{i} \\
0 & \text{otherwise}
\end{cases}
\]

\[A_j^i = \{a_{j1}^i, a_{j2}^i, ..., a_{kj}^i\}\] from the test set \( i \)th feature denotes \( k \) nearest neighbours of \( a_{ji} \). The closer \( a_{ji} \) is to a dense region from the test set, the larger is \( h^*(a_{ji}) \). Thus, it is more reasonable to employ \( h^*(a_{ji}) \) to evaluate the data gravitation. \( h^* \)
\((a_j)\) corresponds to a more flexible range \([0,1]\) than \(b(a_j)\).
Thus, the local data gravitation of instance \(x_j\) can be rewritten as follows:
\[
F_j = w_j^i = \frac{s_j^i}{(m - s_j^i + 1)^r}.
\] (12)

### 3.2 Feature weight

In CFIW-TNB, local data gravitation can be regarded as the weight of training instance. Besides instances are weighted, features are also weighted. In CFWNB [21], the authors argue that the features with maximum mutual relevance and minimum average mutual redundancy are considered to be highly predictive features and thus have greater feature weights. On the basis of CFWNB, our proposed CFIW-TNB considers the distribution distance of the features of two domains.

In our proposed CFIW-TNB, we argue that for predictor highly predictive features should be correlated with the task, yet uncorrelated with other features and minimizing the distance of distribution between source and target projects. We employ mutual information to measure the correlation between each pair of features and Kullback-Leibler divergence (KL) to measure the distribution distance of features between source and target projects. Therefore, the various correlations and distribution distance can be, respectively, written as follows:

\[
I(A_i, C) = \sum_{a_i} P(a_i, c) \log \frac{P(a_i, c)}{P(a_i)P(c)},
\] (13)

\[
I(A_i, A_j) = \sum_{a_i, a_j} P(a_i, a_j) \log \frac{P(a_i, a_j)}{P(a_i)P(a_j)}.
\] (14)

\[
KL(A_i \| A_j) = \sum_{a_i = a_j} P(a_i) \log \frac{P(a_i)}{P(a_j)}.
\] (15)

where \(C\) is the class label, \(A_i\) and \(A_j\) are two different feature variables in the training set; \(A_i\) is the corresponding feature of \(A_i\) in test set; \(c, a_i, a_j, a'_j\), respectively, denote the values that they take. The formula is displayed in Equation (16):

\[
D_i = \frac{NI(A_i, C) - NI(A_i, A'_j)}{m - 1} - \frac{1}{m - 1} \sum_{j=1}^{m} NI(A_i, A_j),
\] (16)

where \(NI(A_i, C)\), \(NI(A_i, A'_j)\) and \(NI(A_i, A_j)\) are normalized \(I(A_i, C)\), \(KL(A_i \| A'_j)\) and \(I(A_i, A_j)\), respectively. Implementing normalization for the calculations is to keep their ranges realistic and consistent. The formulas are illustrated in Equations (17)–(19), respectively.

\[
NI(A_i, C) = \frac{I(A_i, C)}{\sum_{j=1}^{m} I(A_j, C)}.
\] (17)

\[
NI(A_i, A'_j) = \frac{KL(A_i \| A'_j)}{\sum_{j=1}^{m} KL(A_j \| A'_j)}.
\] (18)

\[
NI(A_i, A_j) = \frac{I(A_i, A_j)}{\sum_{j=1}^{m} \sum_{j=1}^{n} I(A_i, A_j)}.
\] (19)

Because \(D_i\) may be a negative number and yet meanwhile the weights for feature are required to be positive, we finally utilize the sigmoid function to transform its value into
the range (0,1). The final weight for feature is written as follows:

\[
w^j_{\text{feature}} = \frac{1}{1 + e^{-dy^j}}. \tag{20}\]

The formula is the same as Equation (1), we repeat weighting feature NBC here for convenience:

\[c(x) = \arg \max P(c) \prod_{i=1}^{m} P(a_i | c)^{w^i_{\text{feature}}}, \tag{21}\]

where prior probability \(P(c)\) and the conditional probability \(P(a_i | c)\) also repeat

\[p(c) = \frac{1 + \sum_{j=1}^{n} w^j_{\text{instance}} \delta(c_j, c)}{q + n}, \tag{22}\]

\[p(a_i | c) = \frac{1 + \sum_{j=1}^{n} w^j_{\text{instance}} \delta(a_{ij}, a_i) \delta(c_j, c)}{\sum_{j=1}^{n} w^j_{\text{instance}} \delta(c_j, c) + n_i}. \tag{23}\]

Now, the detailed Naive Bayes for our proposed CFIW-TNB can be briefly depicted as Algorithm 4. This algorithm includes four parts: (i) computing instance weights; (ii) computing feature weights; (iii) training NBC; and (iv) predicting label of target datasets.

---

**Algorithm 4 CFIW-TNB**

**Input:** training dataset: \(D\), an un-label test dataset: \(L\).

**Output:** test instance prediction label for test cases: \(c(x)\), \(x \in L\).

//Part 1: determine instance weights.

1. Compute Max and Min vectors;
2. for each instance \(x_i \in D\) do
   1. Compute \(s^i_j\) of each instance \(x_i\) by Equation (10);
   2. Compute instance weight \(w^i_{\text{instance}}\) by Equation (12);
3. end for.

//Part 2: determine feature weights.

1. for each pair of features \(A_i, A'_j(i = 1, 2, \ldots, m)\) do
   1. Compute \(I(A_i, C)\) by Equation (13);
   2. Compute \(KL(A_i || A'_j)\) by Equation (15);
   3. Compute \(NI(A_i, C)\) by Equation (17);
   4. Compute \(NI(A_i, A'_j)\) by Equation (18);
2. end for.

//Part 3: training NBC;

1. Compute the prior probability \(P(c)\) by Equation (22);
2. Compute the conditional probability \(P(a_i | c)\) by Equation (23);
3. Build NBC by Equation (21).
4. //Part 4 predict label of target.

1. Predict label \(c(x)\) of \(x \in L\) by using NBC.
2. Return \(c(x)\).

---

## 4 | EXPERIMENTAL SETUP

In this section, we describe the experiments in detail to evaluate the CFIW-TNB, including benchmark datasets, evaluation measures and parameter setting.

### 4.1 | Benchmark dataset

To evaluate the proposed CFIW-TNB, we selected 25 different open-projects from the PROMISE data repository collected by Jureczko [35]; and they are extensively adopted in previous empirical studies [3,4,36]. Each module in these projects contain 20 static code features and a defective label (defective or clean). The instance is labelled as 1 if it includes one or more defects. On the contrary, it is labelled as 0 if it is clean. Table 1 summarizes essential information of these projects including project name, the number of instances and the defective ratios. Table 2 presents the description of features including category, feature name and description.

### 4.2 | Performance measures

In each classification task, the predictive result is one of the following four outcomes [37]:

- True-positive (TP): truly defective instances are predicted as defective;
- False-negative (FN): truly defective instances are predicted as clean;
- False-positive (FP): truly clean instances are predicted as defective;
- True-negative (TN): truly clean instances are predicted as clean.

In our study, we employed the common four measures to evaluate the predictors: probability of detection (PD), AUC, G-measure and F-measure. These measures are defined as follows.

**Recall** is the accuracy rate for predicted truly defective instance. It is hoped that the predictor can identify more defective modules or classes. It is defined as:

\[
\text{recall} = \frac{TP}{TP + FN}. \tag{24}\]
**TABLE 1** Statistics of the PROMISE dataset (defect rate: the percentage of defective instances)

| Dataset | Instances | % Defective |
|---------|-----------|-------------|
| Ant-1.3 | 125       | 16.00%      |
| Ant-1.4 | 178       | 22.47%      |
| Ant-1.5 | 293       | 10.92%      |
| Ant-1.6 | 351       | 26.21%      |
| Ant-1.7 | 745       | 22.28%      |
| Ivy-1.1 | 111       | 56.76%      |
| Ivy-1.4 | 241       | 6.64%       |
| Ivy-2.0 | 342       | 11.36%      |
| Jedit-3.2 | 272 | 33.09%      |
| Jedit-4.0 | 306     | 24.51%      |
| Jedit-4.1 | 312     | 25.32%      |
| Jedit-4.2 | 367     | 13.08%      |
| log4j-1.0 | 135   | 25.19%      |
| log4j-1.1 | 109   | 33.94%      |
| log4j-1.2 | 205   | 92.20%      |
| Poi-1.5 | 237       | 59.49%      |
| Poi-2.0 | 314       | 11.78%      |
| Poi-2.5 | 385       | 64.42%      |
| Poi-3.0 | 442       | 63.57%      |
| Xalan-2.4 | 723 | 15.21%      |
| Xalan-2.5 | 803 | 48.19%      |
| Xalan-2.6 | 885 | 46.44%      |
| Xerces-1.2 | 440 | 16.14%      |
| Xerces-1.3 | 453 | 15.23%      |
| Xerces-1.4 | 588 | 74.32%      |

**TABLE 2** The descriptions of metrics [36]

| Category | Feature | Description |
|----------|---------|-------------|
| Complexity | LOC | Lines of code in a class |
| WMC | Weighted methods per class |
| NPM | Number of public methods |
| AMC | Average method complexity |
| Max_cc | Maximum McCabe's cyclomatic complexity |
| Avg_cc | Average McCabe's cyclomatic complexity |
| MOA | The number of attributes in a class |
| Coupling | CBO | Coupling between object classes |
| RFC | Response for a class |
| CA | The number of other classes that invoke a class. |
| CE | The number of classes invoked in a class. |
| IC | The number of parent classes coupled to a class. |
| CBM | Coupling between methods |
| Cohesion | LCOM | Lack of cohesion in methods |
| LCOM3 | The normalized number of LCOM |
| CAM | Cohesion among methods of class. |
| Abstraction | DIT | Depth of inheritance tree. |
| NOC | The number of public methods in a given class. |
| MFA | Measure of functional abstraction. |
| Encapsulation | DAM | Data access metric. |

\[
 pf = \frac{FP}{FP + TN}. \tag{27}
\]

**G-measure**, combining recall and pf, is defined as

\[
 G - measure = \frac{2 \times recall \times (1 - pf)}{recall + (1 - pf)}. \tag{28}
\]

Receiver operating characteristics (ROC) curve is a two-dimensional graph in which the x-axis and y-axis are recall and pf, respectively. ROC is suitable for evaluating the overall classifier’s performance. AUC calculates the area under the ROC curve, which is independent of the decision threshold (0.5 by default) since AUC in some sense averages performance over all decision thresholds. AUC is an interesting statistics, it is a probability that a randomly positive instance will be ranked higher than a randomly negative instance [20].

**4.3 | Statistical test method**

To further analyse the significant difference between the proposed CFIW-TNB and other predictors, we conducted the non-parametric Friedman-test with the post-hoc Nemenyi’s-
test at the 95% confidence level. The Friedman test assesses whether there are statistically significant differences among different predictors. The statistics of the Friedman-test is written as follows:

$$
\tau_x = \frac{12n}{m(m+1)} \left( \sum_{i=1}^{m} R_i^2 - \frac{m(m+1)^2}{4} \right),
$$

where \( n \) and \( m \), respectively, denote the total number of the datasets and methods, and \( R_i \) denotes the average ranks of the \( i \)-th method. The statistics \( \tau_x \) can be regarded as following the \( \chi^2 \) distribution with \( m-1 \) degree of freedom. And

$$
\tau_F = \frac{(n-1)\tau_x}{m(n-1) - \tau_x},
$$

follows the \( F \)-distribution with \( n-1 \) and \( (m-1)(n-1) \) degrees of freedom. We can compare the corresponding \( p \) the Friedman value, then determine whether or not to accept the null hypothesis that there are no statistically significant differences among all models. If the null hypothesis is rejected, the post-hoc Nemenyi’s-test is performed to check that there is a significant difference between the specific two models. The Nemenyi-test is employed to obtain the critical value of average rank (Critical Difference [CD]) that is calculated with the following formula

$$
CD_{a,m} = q_{a,m} \sqrt{\frac{m(m+1)}{6n}},
$$

where \( q_{a,m} \) is a definite value related to \( m \) and the significant level \( \alpha \), and it is available online.\(^1\) For two predictors, the difference of average rank exceeds the CD, which means to reject the null hypothesis there is no significant difference between them. This significant difference between two predictors can be measured using CD. As shown in Figure 2, two dotted blue lines are shorter than the full red line, which means that predictor 2 is both non-significant difference with predictor 1 and predictor 3. However, the dotted green line is longer than the full red line, which means that predictor 1 and predictor 3 are significantly different. This mean post-hoc Nemenyi’s-test cannot create distinct ranks for the models that are compared, but overlapping ranks [38,39]. In this article, we employed the strategy in Ref. [39] to address this issue, which divides the models into different non-overlapping clusters according to the distance between the best average rank and the worst rank (a green broken line in Figure 2). If this distance is less than 1 CD value they belong to the top rank group; if this distance is two times the CD value, they are divided into three groups: top rank group, middle rank group and bottom rank group; if this distance is less than two times the CD value and larger than one time the CD value, they are divided into two non-overlapping groups: top rank group and bottom rank group. The model belongs to the top rank group (or bottom group) if its average rank is closer to the best average rank (or the worst average rank). If this distance is less than the CD value, all models belong to the same group.

### 4.4 Parameter setting

In CFIW-TNB, calculating local data gravitation for each training instance requires selecting its \( k \) nearest neighbours from the test set. In order to investigate the influence \( k \) (number of nearest neighbours), we change the value of \( k \) based on the number of test instances with different percentages (e.g. \( k = n_{test} \times 20\% \), \( n_{test} \) is the number of test instances). Here, we choose ant-1.4, ivy-1.1, jedit 4.0 and xerces 1.4 to investigate \( k \) values by using 10 cross-validation. Figure 3 shows the average results of CFIW-TNB on \( k \) values with different percentages of the number of test instances. Figure 3 indicates that the predictive results will be affected by \( k \), but this effect is not very sensitive. Therefore, we apply 2% of the number of test instances as the value of \( k \).

### 5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we give the experimental results of CFIW-TNB effectiveness and comparison methods. Based on the results achieved, the research questions have been answered as follows.

**RQ 1** How effective is CFIW-TNB method compared with other CPDP methods?

To explore whether the CFIW-TNB approach could surpass other CPDP approaches, we adopted five existing state-of-the-art CPDP approaches as comparisons. These five CPDP approaches including a CPDP approach designed with a limited amount of labelled within-project data (TrAdaboost); three CPDP approaches using only cross-project data to train predictor (NN filter, TCA+ and TNB); a baseline using an underlying classifier trained by the source project to directly predict the defects of the target project. In the work, we selected logistic regression as the underlying classifier. Before each time experiment, we have to select the source domain for one defect prediction. Specifically, we picked up one release of a project as a target project (e.g. ant-1.3), and used releases of other projects as the candidate source projects (i.e. 20 releases not belonging to the ant in the example).

---

1. [http://www.ccin.ufpe.br/~ac/Nemenyi_critval.pdf](http://www.ccin.ufpe.br/~ac/Nemenyi_critval.pdf)
The experimental results are described in Tables 3 and 4. From the penultimate lines of these tables, CFIW-TNB, respectively, improves TNB, TCA+, NN filter, TrAdaBoost and Baseline by 4.42%, 6.65%, 16.26%, 10.24% and 15.80% in terms of AUC; by 13.36%, 19.71%, 36.69%, 23.38% and 34.23% in terms of F-measure; by 3.36%, 13.09%, 24.45%, 16.07% and 29.96% in terms of pd; by 13.86%, 20.69%, 31.55%, 24.66% and 33.63% in terms of G-measure. The w/t/l means that CFIW-TNB wins in w datasets, ties in t datasets and loses in l datasets. From the fourth line from the end of Table 3, compared with TNB, CFIW-TNB wins 16 times, ties 5 times and loses 4 times in terms of AUC; compared with TCA+, AUC of CFIW-TNB wins 23 times and ties 2 times; compared with NN filter, TrAdaBoost and Baseline, AUC of CFIW-TNB is significantly better on 25 datasets and significantly worse on none. In terms of F-measure, pd and G-measure, the CFIW-TNB approach has won more times than other five CPDP approaches on 25 real-world datasets.

For the four measures (AUC, F-measure, pd and G-measure), we obtained the ranking of all the approaches on each dataset. Figure 4 shows the rankings of four measures for six CPDP approaches using boxplot. Across 25 datasets, the rank of the most frequent occurrence of CFIW-TNB is 1 (ranking number one), the worst ranking is 2 in terms of AUC; the ranking of the most frequent occurrence of CFIW-TNB is 1, the worst ranking is 4 in terms of F-measure, pd and G-measure. Overall, ranks of CFIW-TNB are higher than other CPDP approaches on the four measures in most cases.

For further fair comparisons of the six CPDP approaches, we resorted to the well-known Friedman test with Nemenyi’s test. We could obtain \( q_{0.05,6} = 2.850 \) by querying the certain table. Then the CD value is computed to be \( \text{CD}_{0.05,6} = 2.850 \sqrt{\frac{6 	imes 7}{6 	imes 25}} = 1.5081. \) In detail, the Friedman-test with Nemenyi’s test was analysed in terms of AUC. Based on the average ranks (AR) displayed on the last line of Table 3, we could calculate the statistical variable \( \tau_P = 91.0055. \) For \( \alpha = 0.05, F_{\alpha}(5, 120) = 2.2899 < 91.0055, \) so we rejected the null hypothesis. Then, the Nemenyi’s test was exploited to further compare the six CPDP approaches. We could derive the differences between the AR of CFIW-TNB and that of other CPDP approaches as follows:

\[
d(AR_{\text{Baseline}} - AR_{\text{CFIW-TNB}}) = 5.4 - 1.22 = 4.18 > \text{CD} = 1.5081;
\]
| Dataset | AUC   |  |  |  | F-measure  |  |  |  |  |  |  |
|---------|-------|---|---|---|------------|---|---|---|---|---|---|
|         | AUC   |  |  |  | F-measure  |  |  |  |  |  |  |
|         |       |  |  |  | CFIW-TNB   | TNB | TCA⁺  | NN filter | TrAdaboost | Baseline | CFIW-TNB   | TNB | TCA⁺  | NN filter | TrAdaboost | Baseline |
| Ant-1.3 | 0.784 | 0.704 | 0.647 | 0.632 | 0.671 | 0.614 | 0.349 | 0.321 | 0.351 | 0.318 | 0.331 | 0.321 |
| Ant-1.4 | 0.671 | 0.671 | 0.629 | 0.567 | 0.619 | 0.565 | 0.367 | 0.371 | 0.339 | 0.293 | 0.316 | 0.292 |
| Ant-1.5 | 0.789 | 0.717 | 0.789 | 0.655 | 0.682 | 0.634 | 0.296 | 0.264 | 0.272 | 0.264 | 0.271 | 0.251 |
| Ant-1.6 | 0.733 | 0.734 | 0.654 | 0.624 | 0.678 | 0.642 | 0.581 | 0.534 | 0.477 | 0.359 | 0.459 | 0.414 |
| Ant-1.7 | 0.759 | 0.704 | 0.646 | 0.625 | 0.661 | 0.631 | 0.758 | 0.415 | 0.395 | 0.318 | 0.325 | 0.309 |
| Ivy-1.1 | 0.708 | 0.708 | 0.635 | 0.640 | 0.632 | 0.617 | 0.635 | 0.421 | 0.427 | 0.399 | 0.567 | 0.347 |
| Ivy-1.4 | 0.727 | 0.730 | 0.696 | 0.624 | 0.713 | 0.696 | 0.385 | 0.385 | 0.348 | 0.361 | 0.289 | 0.363 |
| Ivy-2.0 | 0.789 | 0.725 | 0.677 | 0.617 | 0.678 | 0.677 | 0.432 | 0.395 | 0.415 | 0.351 | 0.447 | 0.327 |
| Jedit-3.2 | 0.812 | 0.795 | 0.717 | 0.687 | 0.732 | 0.697 | 0.553 | 0.481 | 0.456 | 0.357 | 0.456 | 0.436 |
| Jedit-4.0 | 0.754 | 0.671 | 0.698 | 0.608 | 0.637 | 0.584 | 0.448 | 0.428 | 0.379 | 0.368 | 0.425 | 0.415 |
| Jedit-4.1 | 0.736 | 0.737 | 0.736 | 0.628 | 0.578 | 0.582 | 0.522 | 0.472 | 0.449 | 0.411 | 0.472 | 0.352 |
| Jedit-4.2 | 0.768 | 0.721 | 0.727 | 0.661 | 0.717 | 0.721 | 0.372 | 0.384 | 0.352 | 0.357 | 0.344 | 0.324 |
| log4j-1.0 | 0.753 | 0.696 | 0.730 | 0.650 | 0.676 | 0.632 | 0.493 | 0.473 | 0.450 | 0.414 | 0.474 | 0.422 |
| log4j-1.1 | 0.714 | 0.714 | 0.708 | 0.620 | 0.687 | 0.662 | 0.504 | 0.503 | 0.488 | 0.412 | 0.425 | 0.405 |
| log4j-1.2 | 0.676 | 0.686 | 0.639 | 0.627 | 0.618 | 0.612 | 0.473 | 0.472 | 0.466 | 0.422 | 0.471 | 0.396 |
| Poi-1.5 | 0.694 | 0.694 | 0.691 | 0.611 | 0.641 | 0.621 | 0.460 | 0.460 | 0.479 | 0.382 | 0.439 | 0.419 |
| Poi-2.0 | 0.729 | 0.673 | 0.709 | 0.627 | 0.638 | 0.632 | 0.267 | 0.267 | 0.256 | 0.215 | 0.243 | 0.193 |
| Poi-2.5 | 0.751 | 0.690 | 0.654 | 0.654 | 0.694 | 0.641 | 0.777 | 0.529 | 0.521 | 0.364 | 0.293 | 0.493 |
| Poi-3.0 | 0.802 | 0.802 | 0.761 | 0.722 | 0.765 | 0.673 | 0.781 | 0.550 | 0.533 | 0.445 | 0.548 | 0.448 |
| Xalan-2.4 | 0.679 | 0.641 | 0.647 | 0.618 | 0.658 | 0.607 | 0.324 | 0.332 | 0.268 | 0.238 | 0.357 | 0.337 |
| Xalan-2.5 | 0.689 | 0.653 | 0.675 | 0.621 | 0.651 | 0.613 | 0.578 | 0.477 | 0.464 | 0.421 | 0.432 | 0.431 |
| Xalan-2.6 | 0.694 | 0.691 | 0.689 | 0.649 | 0.646 | 0.596 | 0.635 | 0.571 | 0.459 | 0.386 | 0.434 | 0.314 |
| xerces1.2 | 0.759 | 0.717 | 0.749 | 0.656 | 0.717 | 0.672 | 0.218 | 0.378 | 0.201 | 0.198 | 0.186 | 0.164 |
| xerces1.3 | 0.744 | 0.691 | 0.726 | 0.641 | 0.662 | 0.691 | 0.428 | 0.367 | 0.392 | 0.386 | 0.441 | 0.445 |
| xerces1.4 | 0.853 | 0.816 | 0.781 | 0.706 | 0.791 | 0.721 | 0.704 | 0.636 | 0.671 | 0.589 | 0.557 | 0.575 |
| w/t/l | – | 16/5/4 | 23/2/0 | 25/0/0 | 25/0/0 | 25/0/0 | – | 18/3/4 | 23/0/2 | 25/0/0 | 22/0/3 | 23/0/2 |
| Average | 0.743 | 0.711 | 0.696 | 0.639 | 0.674 | 0.641 | 0.494 | 0.435 | 0.412 | 0.361 | 0.400 | 0.368 |
| Improved | – | 4.42% | 6.65% | 16.26% | 10.24% | 15.80% | – | 13.36% | 19.71% | 36.69% | 23.38% | 34.23% |
| AR | 1.22 | 2.24 | 3.06 | 5.28 | 3.8 | 5.4 | 1.44 | 2.56 | 3.24 | 5.14 | 3.600 | 5.02 |
| Dataset       | pd   | G-measure   |
|--------------|------|-------------|
|               | CFIW-TNB | TNB | TCA+ | NN filter | TrAdaboost | Baseline | CFIW-TNB | TNB | TCA+ | NN filter | TrAdaboost | Baseline |
| Ant-1.3       | 0.942 | 0.903 | 0.835 | 0.683 | 0.822 | 0.705 | 0.428 | 0.368 | 0.387 | 0.381 | 0.376 | 0.372 |
| Ant-1.4       | 0.912 | 0.912 | 0.814 | 0.673 | 0.793 | 0.672 | 0.455 | 0.427 | 0.384 | 0.328 | 0.367 | 0.322 |
| Ant-1.5       | 0.884 | 0.813 | 0.725 | 0.656 | 0.766 | 0.665 | 0.637 | 0.390 | 0.331 | 0.282 | 0.310 | 0.260 |
| Ant-1.6       | 0.886 | 0.765 | 0.739 | 0.649 | 0.691 | 0.635 | 0.655 | 0.586 | 0.504 | 0.423 | 0.484 | 0.412 |
| Ant-1.7       | 0.842 | 0.842 | 0.753 | 0.644 | 0.632 | 0.625 | 0.622 | 0.439 | 0.413 | 0.352 | 0.351 | 0.336 |
| Ivy-1.1       | 0.876 | 0.805 | 0.767 | 0.693 | 0.647 | 0.583 | 0.658 | 0.471 | 0.485 | 0.467 | 0.616 | 0.455 |
| Ivy-1.4       | 0.901 | 0.808 | 0.761 | 0.662 | 0.742 | 0.621 | 0.495 | 0.394 | 0.421 | 0.405 | 0.326 | 0.419 |
| Ivy-2.0       | 0.837 | 0.807 | 0.735 | 0.631 | 0.737 | 0.669 | 0.483 | 0.435 | 0.451 | 0.397 | 0.488 | 0.454 |
| Jedit-3.2     | 0.773 | 0.786 | 0.709 | 0.612 | 0.732 | 0.654 | 0.634 | 0.490 | 0.504 | 0.427 | 0.492 | 0.455 |
| Jedit-4.0     | 0.735 | 0.727 | 0.689 | 0.685 | 0.707 | 0.647 | 0.492 | 0.459 | 0.411 | 0.442 | 0.494 | 0.457 |
| Jedit-4.1     | 0.697 | 0.708 | 0.669 | 0.727 | 0.682 | 0.652 | 0.591 | 0.512 | 0.487 | 0.657 | 0.491 | 0.482 |
| Jedit-4.2     | 0.759 | 0.679 | 0.649 | 0.701 | 0.657 | 0.657 | 0.426 | 0.426 | 0.374 | 0.399 | 0.386 | 0.380 |
| log4j-1.0     | 0.629 | 0.670 | 0.629 | 0.571 | 0.598 | 0.562 | 0.545 | 0.511 | 0.489 | 0.446 | 0.519 | 0.502 |
| log4j-1.1     | 0.583 | 0.618 | 0.609 | 0.473 | 0.601 | 0.398 | 0.565 | 0.573 | 0.525 | 0.445 | 0.493 | 0.497 |
| log4j-1.2     | 0.587 | 0.613 | 0.589 | 0.475 | 0.573 | 0.472 | 0.522 | 0.476 | 0.533 | 0.461 | 0.481 | 0.479 |
| Poi-1.5       | 0.634 | 0.619 | 0.555 | 0.434 | 0.538 | 0.414 | 0.479 | 0.513 | 0.521 | 0.428 | 0.482 | 0.410 |
| Poi-2.0       | 0.553 | 0.586 | 0.521 | 0.493 | 0.503 | 0.416 | 0.323 | 0.323 | 0.286 | 0.287 | 0.275 | 0.248 |
| Poi-2.5       | 0.472 | 0.593 | 0.487 | 0.432 | 0.468 | 0.385 | 0.653 | 0.579 | 0.533 | 0.416 | 0.360 | 0.521 |
| Poi-3.0       | 0.621 | 0.580 | 0.453 | 0.487 | 0.526 | 0.464 | 0.736 | 0.693 | 0.648 | 0.577 | 0.556 | 0.530 |
| Xalan-2.4     | 0.597 | 0.639 | 0.612 | 0.567 | 0.592 | 0.569 | 0.375 | 0.357 | 0.295 | 0.379 | 0.403 | 0.322 |
| Xalan-2.5     | 0.673 | 0.708 | 0.691 | 0.647 | 0.638 | 0.285 | 0.655 | 0.609 | 0.501 | 0.428 | 0.498 | 0.464 |
| Xalan-2.6     | 0.779 | 0.687 | 0.710 | 0.727 | 0.684 | 0.703 | 0.651 | 0.493 | 0.512 | 0.443 | 0.441 | 0.457 |
| xerces1.2     | 0.741 | 0.574 | 0.612 | 0.547 | 0.553 | 0.593 | 0.384 | 0.424 | 0.250 | 0.245 | 0.205 | 0.217 |
| xerces1.3     | 0.753 | 0.661 | 0.514 | 0.452 | 0.545 | 0.535 | 0.485 | 0.428 | 0.409 | 0.395 | 0.527 | 0.454 |
| xerces1.4     | 0.995 | 0.952 | 0.674 | 0.674 | 0.651 | 0.778 | 0.790 | 0.691 | 0.742 | 0.543 | 0.613 | 0.445 |
| w/t/l         | – | 14/2/9 | 19/1/5 | 24/0/1 | 24/0/1 | 25/0/0 | – | 20/2/3 | 22/0/3 | 23/0/2 | 20/0/5 | 25/0/0 |
| Average       | 0.746 | 0.722 | 0.660 | 0.600 | 0.643 | 0.574 | 0.550 | 0.483 | 0.456 | 0.418 | 0.441 | 0.412 |
| Improved      | – | 3.36% | 13.09% | 24.45% | 16.07% | 29.90% | – | 13.86% | 20.69% | 31.55% | 24.66% | 33.63% |
| AR            | 1.68 | 1.88 | 3.36 | 4.72 | 4.04 | 5.32 | 1.48 | 2.96 | 3.32 | 4.56 | 3.68 | 5 |
\[ d(AR_{\text{TrAdaboost}} - AR_{\text{CFIW-TNB}}) = 3.8-1.22 = 2.58 \]
\[ CD = 1.5081; \]
\[ d(AR_{\text{NNfilter}} - AR_{\text{CFIW-TNB}}) = 5.28-1.22 = 4.06 \]
\[ CD = 1.5081; \]
\[ d(AR_{\text{TCA+}} - AR_{\text{CFIW-TNB}}) = 3.06-1.22 = 1.84 \]
\[ CD = 1.5081; \]
\[ d(AR_{\text{TNB}} - AR_{\text{CFIW-TNB}}) = 2.24-1.22 = 1.02 \]
\[ CD = 1.5081; \]

where \(d(a)-(b)\) denotes the difference between AR of two approaches (a) and (b). We conclude that CFIW-TNB performs significantly better than Baseline, NN filter and TCA+. The distances between AR of CFIW-TNB and TNB, between AR of TCA+ and TNB are both less than CD, which indicates that there is no significant difference between TNB and TCA+, TNB and CFIW-TNB. The Nemenyi’s test generated overlapping groups for approaches. In this work, we utilized the strategy in Ref. [40] to address this issue. The distance between CFIW-TNB and TCA+ is less than two times the CD and more than one time of the CD. The AR of TNB is closer than that of TCA+ so they belong to the same rank group. Similarly, we could obtain the same conclusions in terms of other measures based on the average ranks and relevant values reported in Table 5. Figure 5 shows the results of the Friedman test with Nemenyi’s test for six CPDP approaches in terms of the four measures. Groups of the approaches that are significantly different are with different colours. The results of the post-hoc test indicate that CFIW-TNB always belongs to the top rank group in terms of the four measures.

**RQ 2 Is dual weighting mechanism helpful for CPDP?**

To explore whether the dual weights mechanism can improve the performance of CPDP, we adopted three existing NBC, including TNB [13], CFW-NB [21] and Naive Bayes (NB), as comparisons. TNB and CFW-TB are single instance weighting NBC and single feature weighting NBC, respectively. CFW-NB and NB classifiers trained by source domain to directly predict the defects of target protect. For TNB, we repeated the above experiment on TNB.

The experimental results are described in Tables 6 and 7. From the penultimate lines of this table, CFIW-TNB, respectively, improves TNB, CFW-NB and NB by 4.42%, 9.38% and 15.0% in terms of AUC; by 13.36%, 16.49% and 32.72% in terms of F-measure; by 3.36%, 14.42% and 26.83% in terms of \(p_d\); by 13.97%, 23.23% and 32.62% in terms of G-measure. Compared with TNB, CFW-NB and NB, CFIW-TNB is significantly better on 16, 24 and 25 datasets in terms of AUC, respectively; compared with TNB, CFW-NB and NB, CFIW-TNB is significantly more outstanding on 18, 20 and 22 datasets in terms of F-measure, respectively; compared with TNB, CFW-NB and NB, CFIW-TNB is significantly more accurate on 14, 22 and 28 datasets in terms of \(p_d\), respectively; compared with TNB, CFW-NB and NB, CFIW-TNB is significantly more excellent on 20, 24 and 25 datasets in terms of G-measure, respectively.

Figure 6 depicts the box-plots of four measures for the four NBC across all 25 datasets. Figure 6 shows the superiority of our CFIW-TNB compared with the three NBC in terms of the median values of all four measures across all 25 datasets. To further fairly with these approaches, we resorted to the well-known Friedman test with Nemenyi’s hoc test. We could obtain \(q_{0.05,4} = 2.569\) by querying the certain table. The differences of AR between CFIW-TNB and other NBC and
TABLE 5 Relevant values in the Friedman test with Nemenyi’s post-hoc test on the employed datasets

| Measure  | $\tau_F$ | d(TNB−CIFW-TNB) | d(TCA+−CIFW-TNB) | d(NN filter−CIFW-TNB) | d(TrAdaBoost−CIFW-TNB) | d(Baseline−CIFW-TNB) |
|----------|----------|-----------------|-----------------|-----------------------|------------------------|-----------------------|
| AUC      | 91.0055  | 1.02            | 1.84            | 4.06                  | 2.58                   | 4.18                  |
| F-measure| 33.5721  | 1.12            | 1.80            | 3.70                  | 2.16                   | 3.58                  |
| Pd       | 41.1042  | 0.20            | 1.68            | 3.04                  | 2.36                   | 3.64                  |
| G-measure| 19.3345  | 1.48            | 1.84            | 3.08                  | 2.20                   | 3.52                  |

FIGURE 5 Comparison of six CPDP approaches using Friedman-test with Nemenyi’s-test in terms of the four measures

statistical variables are displayed in Table 8. In terms of four measures, statistical variables $\tau_F$ are greater than $F_{p}(3, 72) = 2.7318$, which indicates that the null hypothesis is rejected. Since differences between CIFW-TNB and CFW-NB, CIFW-TNB and NB are greater than that between CD, the proposed CIFW-TNB performs significantly better than CFW-NB and NB.

Figure 7 shows the results of the Friedman test with Nemenyi’s test in terms of the four measures, which showed that CIFW-TNB belongs to top group in terms of four measures. These observations showed that CIFW-TNB with the dual weighting mechanism performs better than TNB with instance weights and CFW-TNB with feature weights. In summary, we conclude that the performance of the dual weighting mechanism is better than the single weighting mechanism. This is because CIFW-TNB pays different attention to the features and instances with predictive difference, rather than considering a single predictive difference.

RQ 3 Is CIFW-TNB superior to its variants?

Since two weights (feature weight and instance weight) are used in our method, this question is designed to investigate whether or not our method is more effective than single weight or non-weight. To answer this question, we designed three of its variants, including CFW-TNB (only retaining feature weight), CIW-TNB (only retaining feature weight) and NB (non-weight). They can be used to compare the performance of our method against its downgraded version methods that do not consider two weights. All these baseline methods are treated as the variants of CIFW-TNB.

The experimental results are described in Tables 9 and 10. From the penultimate lines of these table, CIFW-TNB, respectively, improves CIW-TNB, CFW-TNB and NB by 4.55%, 5.88% and 15.0% in terms of AUC; by 8.62%, 11.66% and 32.72% in terms of F-measure; by 3.627%, 5.92% and 26.83% in terms of pd; by 8.62%, 10.82% and 32.62% in terms of G-measure.
Compared with CIW-TNB, CFW-NB and NB, CIFW-TNB is more outstanding on 24, 25 and 25 datasets in terms of AUC, respectively; compared with CIW-TNB, CFW-NB and NB, CIFW-TNB is more outstanding on 24, 25 and 22 datasets in terms of F-measure, respectively; compared with CIW-TNB, CFW-NB and NB, CIFW-TNB is more outstanding on 23, 25 and 25 datasets in terms of \(pd\), respectively; compared with CIW-TNB, CFW-NB and NB, CIFW-TNB is more outstanding on 24, 24 and 24 datasets in terms of G-measure, respectively.

Figure 8 depicts the box-plots of four measures for the four NBC across all 25 datasets. Figure 8 shows the superiority of our CIFW-TNB compared with its variants in terms of the median values of all four measures across all 25 datasets. To further fairly with these approaches, we resorted to the well-known Friedman-test with Nemenyi’s hoc test. The differences of AR between CIFW-TNB and its variants and statistical variable are displayed in Table 11. In terms of four measures, statistical variables \(\tau_F\) are greater than \(F_{\alpha}(3, 72) = 2.7318\), which indicates that the null hypothesis is rejected. Since the differences between CIFW-TNB and CFW-TNB, CIFW-TNB and CIW-TNB, CIFW-TNB and NB are greater than that between CD, the proposed CIFW-TNB performs significantly better than its variants.

| Dataset | AUC | F-measure |
|---------|-----|-----------|
|         | CIFW-TNB | TNB | CFW-NB | NB | CIFW-TNB | TNB | CFW-NB | NB |
| Ant-1.3 | 0.784 | 0.704 | 0.709 | 0.648 | 0.349 | 0.321 | 0.347 | 0.354 |
| Ant-1.4 | 0.671 | 0.671 | 0.591 | 0.594 | 0.367 | 0.371 | 0.317 | 0.202 |
| Ant-1.5 | 0.789 | 0.717 | 0.671 | 0.651 | 0.296 | 0.264 | 0.245 | 0.223 |
| Ant-1.6 | 0.733 | 0.734 | 0.657 | 0.675 | 0.581 | 0.534 | 0.514 | 0.423 |
| Ant-1.7 | 0.759 | 0.704 | 0.715 | 0.655 | 0.758 | 0.415 | 0.406 | 0.496 |
| Ivy-1.1 | 0.708 | 0.708 | 0.651 | 0.629 | 0.635 | 0.421 | 0.367 | 0.345 |
| Ivy-1.4 | 0.727 | 0.730 | 0.671 | 0.648 | 0.385 | 0.385 | 0.353 | 0.403 |
| Ivy-2.0 | 0.789 | 0.725 | 0.684 | 0.682 | 0.432 | 0.395 | 0.341 | 0.314 |
| Jedit-3.2 | 0.812 | 0.795 | 0.715 | 0.790 | 0.553 | 0.481 | 0.527 | 0.437 |
| Jedit-4.0 | 0.754 | 0.671 | 0.657 | 0.593 | 0.448 | 0.428 | 0.435 | 0.322 |
| Jedit-4.1 | 0.736 | 0.737 | 0.718 | 0.641 | 0.522 | 0.472 | 0.452 | 0.439 |
| Jedit-4.2 | 0.768 | 0.721 | 0.683 | 0.690 | 0.372 | 0.384 | 0.395 | 0.275 |
| log4j-1.0 | 0.753 | 0.696 | 0.652 | 0.681 | 0.493 | 0.473 | 0.419 | 0.450 |
| log4j-1.1 | 0.714 | 0.714 | 0.694 | 0.651 | 0.504 | 0.503 | 0.485 | 0.356 |
| log4j-1.2 | 0.676 | 0.686 | 0.670 | 0.630 | 0.473 | 0.472 | 0.422 | 0.396 |
| Poi-1.5 | 0.694 | 0.694 | 0.649 | 0.653 | 0.460 | 0.460 | 0.506 | 0.366 |
| Poi-2.0 | 0.729 | 0.673 | 0.616 | 0.576 | 0.267 | 0.267 | 0.313 | 0.168 |
| Poi-2.5 | 0.751 | 0.690 | 0.638 | 0.675 | 0.777 | 0.529 | 0.525 | 0.514 |
| Poi-3.0 | 0.802 | 0.802 | 0.716 | 0.647 | 0.781 | 0.550 | 0.516 | 0.502 |
| Xalan-2.4 | 0.679 | 0.641 | 0.675 | 0.545 | 0.324 | 0.332 | 0.353 | 0.314 |
| Xalan-2.5 | 0.689 | 0.653 | 0.676 | 0.618 | 0.578 | 0.477 | 0.523 | 0.365 |
| Xalan-2.6 | 0.694 | 0.691 | 0.668 | 0.675 | 0.635 | 0.571 | 0.517 | 0.419 |
| xerces1.2 | 0.759 | 0.717 | 0.760 | 0.647 | 0.218 | 0.378 | 0.353 | 0.295 |
| xerces1.3 | 0.744 | 0.691 | 0.642 | 0.619 | 0.428 | 0.367 | 0.413 | 0.392 |
| xerces1.4 | 0.853 | 0.816 | 0.798 | 0.631 | 0.704 | 0.636 | 0.553 | 0.529 |
| \(\omega/t/l\) | – | 16/5/4 | 24/0/1 | 25/0/0 | – | 18/3/4 | 20/0/5 | 22/0/3 |
| Average | 0.743 | 0.711 | 0.679 | 0.646 | 0.494 | 0.435 | 0.424 | 0.372 |
| Improved | – | 4.42% | 9.38% | 15.00% | – | 13.36% | 16.49% | 32.72% |
| AR | 1.22 | 2.02 | 3.08 | 3.68 | 1.48 | 2.36 | 2.6 | 3.56 |
Figure 9 visualized the results of the Friedman test with Nemenyi's test in terms of the four measures for CIFW-TNB and its variants, which showed that CIFW-TNB belongs to top group in terms of four measures. These observations showed that CIFW-TNB with dual weighting mechanism performs better than single weight (only instance or feature weight) and non-weight. In summary, we conclude that the effect of weighting both instances and features is better than that of only weighting instances or features and non-weighting.

6 | THREATS TO VALIDITY

Threats to external validity means the degree to generalize the research results. In this work, a related threat to validity might be triggered by the single data repository (i.e., PROMISE repository). Although CIFW-TNB can obtain good performance on 25 datasets, it is not certain that CIFW-TNB will be completely suitable for other datasets. In addition, since the features of the datasets employed are all static product metrics and the modules are extracted at the class level, we cannot
FIGURE 6   Box-plots of four measures for CIFW-TNB and three NBC across all 25 datasets.

TABLE 8   Relevant values in the Friedman test with Nemenyi’s post-hoc test on the 25 datasets for four NBC approaches

| Measure | $\tau_F$ | CD_{0.05,4} | $F$ (3, 72) | $d$(TNB–CIFW-TNB) | $d$(CFW-NB–CIFW-TNB) | $d$(NB–CIFW-TNB) |
|---------|---------|-------------|-------------|---------------------|----------------------|------------------|
| AUC     | 61.5676 | 0.9379      | 2.7318      | 0.80                | 1.86                 | 2.46             |
| F-measure | 18.7594 | 0.9379      | 2.7318      | 0.88                | 1.12                 | 2.08             |
| $pd$    | 30.2888 | 0.9379      | 2.7318      | 0.44                | 1.42                 | 2.14             |
| G-measure | 179.8043 | 0.9379     | 2.7318      | 0.84                | 1.68                 | 2.84             |

FIGURE 7   Friedman test with Nemenyi’s test in terms of the four measures for CIFW-TNB and three NBC across all 25 datasets.
guarantee that our approach can be generalised to the defect datasets with process metrics and the modules extracted at the file level. Our approach should encourage more researchers to develop more practical prediction models.

Threats to construct validity mainly refer to selecting the baselines and the evaluative measures. Totally five CPDP approaches and three NBCs are chosen as comparisons. The authors cannot compare CFIW-TNB with all existing CPDP approaches and NBC with instance weighting or feature weighting since the space is limited. We employ four extensively used measures to evaluate the predictive performances of our CFIW-TNB and comparisons for defect prediction. These measures do not take classification costs, different classification errors result in different costs, into consideration, which may be a potential threat to the validity of results.

### TABLE 9 The AUC and F-measure of our CFIW-TNB compared with CIW-TNB CFW-NB and NB, respectively

| Dataset | AUC | F-measure |
|---------|-----|-----------|
|         | AUC | CIW-TNB | CFW-TNB | NB | AUC | CIW-TNB | CFW-TNB | NB |
| Ant-1.3 | 0.784 | 0.714 | 0.739 | 0.648 | 0.349 | 0.331 | 0.341 | 0.354 |
| Ant-1.4 | 0.671 | 0.637 | 0.654 | 0.594 | 0.367 | 0.362 | 0.317 | 0.202 |
| Ant-1.5 | 0.789 | 0.747 | 0.732 | 0.651 | 0.296 | 0.274 | 0.245 | 0.223 |
| Ant-1.6 | 0.733 | 0.721 | 0.712 | 0.675 | 0.581 | 0.562 | 0.514 | 0.423 |
| Ant-1.7 | 0.759 | 0.731 | 0.725 | 0.655 | 0.758 | 0.615 | 0.606 | 0.496 |
| Ivy-1.1 | 0.708 | 0.692 | 0.701 | 0.629 | 0.635 | 0.422 | 0.367 | 0.345 |
| Ivy-1.4 | 0.727 | 0.713 | 0.715 | 0.648 | 0.385 | 0.382 | 0.363 | 0.403 |
| Ivy-2.0 | 0.789 | 0.745 | 0.753 | 0.682 | 0.432 | 0.405 | 0.381 | 0.314 |
| Jedit-3.2 | 0.812 | 0.815 | 0.782 | 0.79 | 0.553 | 0.511 | 0.532 | 0.437 |
| Jedit-4.0 | 0.754 | 0.651 | 0.656 | 0.593 | 0.448 | 0.421 | 0.436 | 0.322 |
| Jedit-4.1 | 0.736 | 0.721 | 0.711 | 0.641 | 0.522 | 0.512 | 0.518 | 0.439 |
| Jedit-4.2 | 0.768 | 0.742 | 0.681 | 0.69 | 0.372 | 0.364 | 0.345 | 0.275 |
| log4j-1.0 | 0.753 | 0.716 | 0.721 | 0.681 | 0.493 | 0.476 | 0.489 | 0.450 |
| log4j-1.1 | 0.714 | 0.693 | 0.697 | 0.651 | 0.504 | 0.503 | 0.492 | 0.356 |
| log4j-1.2 | 0.676 | 0.625 | 0.637 | 0.63 | 0.473 | 0.438 | 0.452 | 0.396 |
| Poi-1.5 | 0.694 | 0.663 | 0.671 | 0.653 | 0.460 | 0.426 | 0.422 | 0.366 |
| Poi-2.0 | 0.729 | 0.687 | 0.696 | 0.576 | 0.267 | 0.241 | 0.243 | 0.168 |
| Poi-2.5 | 0.751 | 0.684 | 0.618 | 0.675 | 0.777 | 0.729 | 0.731 | 0.514 |
| Poi-3.0 | 0.802 | 0.732 | 0.726 | 0.647 | 0.781 | 0.755 | 0.716 | 0.502 |
| Xalan-2.4 | 0.679 | 0.654 | 0.625 | 0.545 | 0.324 | 0.312 | 0.323 | 0.314 |
| Xalan-2.5 | 0.689 | 0.683 | 0.678 | 0.618 | 0.578 | 0.477 | 0.523 | 0.365 |
| Xalan-2.6 | 0.694 | 0.692 | 0.684 | 0.675 | 0.635 | 0.571 | 0.517 | 0.419 |
| xerces1.2 | 0.759 | 0.723 | 0.732 | 0.647 | 0.218 | 0.219 | 0.204 | 0.295 |
| xerces1.3 | 0.744 | 0.742 | 0.692 | 0.619 | 0.428 | 0.417 | 0.421 | 0.392 |
| xerces1.4 | 0.853 | 0.836 | 0.798 | 0.631 | 0.704 | 0.636 | 0.553 | 0.529 |
| W/t/l | – | 24/0/1 | 25/0/0 | 25/0/0 | – | 24/0/1 | 25/0/0 | 22/0/3 |
| Average | 0.743 | 0.710 | 0.701 | 0.646 | 0.494 | 0.454 | 0.442 | 0.372 |
| Improved | – | 4.55% | 5.88% | 15.00% | – | 8.62% | 11.66% | 32.72% |
| AR | 1.04 | 2.48 | 2.64 | 3.84 | 1.16 | 2.56 | 2.68 | 3.60 |

7 | CONCLUSIONS AND FUTURE WORKS

In this work, we propose a new CPDP model with a dual weighting mechanism called CFIW-TNB based on NBC, which is not the prior instance filter and FS transfer learning, but weights for all instances and features. The CFIW-TNB defines a novel local data gravitation as the instance weight to emphasise the impact of instances on source data to target.
Compared with TNB, CFIW-TNB considers the impact of density of data distribution on data gravitation. The denser the data distribution, the greater is the gravitation, yet the sparser the data distribution, the less is the gravitation. Therefore, it is more flexible in allocating weights of instance than TNB. The CFIW-TNB employs feature weights to emphasise more on highly predictive features than those that are less predictive. We argue for a predictor that highly predictive features should be highly correlated with task, uncorrelated with other features and minimized distribution distance between target feature. And they are incorporated into weights of feature. We validate the proposed CFIW-TNB against existing five state-of-the-art CPDP approaches using 25 datasets from the PROMISE data repository, use AUC, F-measure, pd and G-measure as performance measure of predictors. And the non-parametric Friedman-test with the post-hoc Nemenyi-test is taken as the statistical method. Extensive experimental results demonstrate that our CFIW-TNB achieves better results compared with existing CPDP in most cases; and the dual weighting mechanism can improve the performance of predictor better than single instance weighting or feature weighting.

In the future, the authors plan to evaluate CFIW-TNB with datasets from more software projects collected from different repositories. Furthermore, considering a more practical scenario, the authors will extend the proposed CFIW-TNB into imbalance learning and cost-sensitive learning in CPDP tasks.

| Dataset | pd | G-measure |
|---------|-----|-----------|
|         | CIW-TNB | TNB | CFW-NB | NB | CIW-TNB | TNB | CFW-NB | NB |
| Ant-1.3 | 0.942 | 0.915 | 0.896 | 0.728 | 0.428 | 0.388 | 0.403 | 0.279 |
| Ant-1.4 | 0.912 | 0.892 | 0.894 | 0.729 | 0.455 | 0.437 | 0.442 | 0.39 |
| Ant-1.5 | 0.884 | 0.853 | 0.873 | 0.712 | 0.637 | 0.569 | 0.575 | 0.331 |
| Ant-1.6 | 0.886 | 0.837 | 0.825 | 0.581 | 0.655 | 0.616 | 0.582 | 0.569 |
| Ant-1.7 | 0.842 | 0.831 | 0.838 | 0.618 | 0.622 | 0.549 | 0.605 | 0.396 |
| Ivy-1.1 | 0.876 | 0.825 | 0.776 | 0.625 | 0.658 | 0.573 | 0.582 | 0.408 |
| Ivy-1.4 | 0.901 | 0.885 | 0.795 | 0.645 | 0.495 | 0.454 | 0.438 | 0.3 |
| Ivy-2.0 | 0.837 | 0.787 | 0.675 | 0.62 | 0.483 | 0.456 | 0.466 | 0.43 |
| Jedit-3.2 | 0.773 | 0.716 | 0.728 | 0.727 | 0.634 | 0.534 | 0.543 | 0.4 |
| Jedit-4.0 | 0.735 | 0.712 | 0.685 | 0.629 | 0.492 | 0.452 | 0.414 | 0.375 |
| Jedit-4.1 | 0.697 | 0.686 | 0.695 | 0.61 | 0.591 | 0.524 | 0.523 | 0.422 |
| Jedit-4.2 | 0.759 | 0.694 | 0.723 | 0.737 | 0.426 | 0.416 | 0.408 | 0.39 |
| log4j-1.0 | 0.629 | 0.623 | 0.589 | 0.572 | 0.545 | 0.531 | 0.492 | 0.428 |
| log4j-1.1 | 0.583 | 0.538 | 0.544 | 0.462 | 0.565 | 0.532 | 0.527 | 0.511 |
| log4j-1.2 | 0.587 | 0.603 | 0.521 | 0.443 | 0.522 | 0.496 | 0.472 | 0.433 |
| Poi-1.5 | 0.634 | 0.619 | 0.624 | 0.386 | 0.479 | 0.513 | 0.474 | 0.444 |
| Poi-2.0 | 0.553 | 0.521 | 0.484 | 0.413 | 0.323 | 0.322 | 0.293 | 0.234 |
| Poi-2.5 | 0.472 | 0.453 | 0.464 | 0.46 | 0.653 | 0.579 | 0.551 | 0.508 |
| Poi-3.0 | 0.621 | 0.618 | 0.611 | 0.492 | 0.736 | 0.693 | 0.664 | 0.61 |
| Xalan-2.4 | 0.597 | 0.565 | 0.574 | 0.518 | 0.364 | 0.346 | 0.372 | 0.304 |
| Xalan-2.5 | 0.673 | 0.678 | 0.632 | 0.527 | 0.655 | 0.621 | 0.597 | 0.552 |
| Xalan-2.6 | 0.779 | 0.776 | 0.775 | 0.644 | 0.651 | 0.515 | 0.645 | 0.385 |
| xerces1.2 | 0.741 | 0.694 | 0.668 | 0.65 | 0.384 | 0.354 | 0.381 | 0.376 |
| xerces1.3 | 0.753 | 0.712 | 0.752 | 0.595 | 0.485 | 0.447 | 0.321 | 0.32 |
| xerces1.4 | 0.995 | 0.957 | 0.966 | 0.591 | 0.700 | 0.721 | 0.618 | 0.573 |
| w/t/l | – | 23/0/2 | 25/0/0 | 25/0/0 | – | 24/0/1 | 24/0/1 | 25/0/0 |
| Average | 0.746 | 0.720 | 0.7043 | 0.689 | 0.550 | 0.505 | 0.495 | 0.415 |
| Improved | – | 3.67% | 5.92% | 26.83% | – | 8.62% | 10.82% | 32.62% |
| AR | 1.08 | 2.52 | 2.56 | 3.84 | 1.08 | 2.36 | 2.56 | 3.96 |
TABLE 11 Relevant values in the Friedman test with Nemenyi's on the 25 datasets for CIFW-TNB and its variants

| Measure  | $\tau_F$ | $CD_{0.05,4}$ | $F(3, 72)$ | $d(CIFW-TNB-CIFW-TNB)$ | $d(CIFW-TNB-CFW-TNB)$ | $d(NB-CIFW-TNB)$ |
|----------|----------|----------------|------------|------------------------|------------------------|--------------------|
| AUC      | 38.6223  | 0.9379         | 2.7318     | 1.44                   | 1.60                   | 2.80               |
| F-measure| 21.7361  | 0.9379         | 2.7318     | 1.40                   | 1.52                   | 2.44               |
| Pd       | 35.2400  | 0.9379         | 2.7318     | 1.44                   | 1.48                   | 2.76               |
| G-measure| 41.6185  | 0.9379         | 2.7318     | 1.28                   | 1.48                   | 2.88               |

FIGURE 8 Box-plots of four measures for CIFW-TNB and its variants across all 25 datasets

FIGURE 9 Friedman-test with Nemenyi's-test of four measures for CIFW-TNB and its variants across all 25 datasets
ACKNOWLEDGMENTS

We thank Jaechang Nam and Jiang for providing us the source code of TCA and CFW-NB. This work is supported in part by the National Nature Science Foundation of China (No. 61,370,103), the Guangzhou Produce & Research Fund (201,902,020,004) and the Fundamental Research Funds for the Central Universities.

ORCID
Quanyi Zou https://orcid.org/0000-0001-6543-2224

REFERENCES

1. Wang, S., Yao, X.: Using class imbalance learning for software defect prediction’. IEEE Trans. Reliab. 62(2), 434–443 (2013)
2. Ambros, M.D., Lanza, M, Robbes, R.: Evaluating defect prediction approaches: a benchmark and an extensive comparison’. Empirical Sofw. Eng., 17(4/5), 531–577 (2012)
3. Xu, Z., et al.: Software defect prediction based on kernel PCA and weighted extreme learning machine. Inf. Sofw. Technol. 106, 182–200 (2019)
4. Tong, H., Liu, B., Wang, S.: Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning’. Inf. Sofw. Technol. 96, 94–111 (2018)
5. Nam, J., et al.: Heterogeneous defect prediction’. IEEE Trans. Sofw. Eng. 44(9), 874–896 (2018)
6. Pan, S.J., et al.: Domain adaptation via transfer component analysis. IEEE Trans. Neural Netw. 22(2), 199–210 (2011)
7. Qiu, S., Lu, L., Jiang, S.: Multiple-components weights model for cross-project software defect prediction. IET Sofw. 12(4), 345–355 (2018)
8. Pan, S.J., Yang, Q.: A survey on transfer learning’. IEEE Trans. Knowl. Data Eng. 22(10), 1345–1359 (2010)
9. Shi, Y., et al.: Extending semi-supervised learning methods for inductive transfer learning. In: Proceedings of Ninth IEEE International Conference on Data Mining, pp. 483–492, Miami, (2009). https://doi.org/10.1109/ICDM.2009.75
10. Turhan, B.; et al.: On the relative value of cross company and within-company data for defect prediction. Empirical Sofw. Eng. 14(5), 540–578 (2009)
11. Kawata, K., Amasaki, S., Yokogawa, T.: Improving relevancy filter methods for cross-project defect prediction. In: Proceedings of 3rd International Conference on Computing Applications and Information Technology/2nd International Conference on Computational Science and Intelligence, Okayama, (2015)
12. Peters, F., Menzies, T., Marcus, A.: Better cross company defect prediction. In: Proceedings on 10th IEEE International Workshop on Mining Software and Repositories (2013). https://doi.org/10.1109/MSR.2013.6624057
13. Ma, Y., et al.: Transfer learning for cross-company software defect prediction’. Inf. Sofw. Technol. 54(3), 248–256 (2012)
14. Liu, C., et al.: A two-phase transfer learning model for cross-project defect prediction’. Inf. Sofw. Technol. 107, 125–136 (2019)
15. Nam, J., Pan, S.J., Kim, S.: Transfer defect learning. In: Proceedings of 35th International Conference on Software Engineering, San Francisco (2013). https://doi.org/10.1109/ICSE.2013.6606584
16. Yu, Q., et al.: An empirical study on the effectiveness of feature selection for cross-project defect prediction. IEEE Access, 7, 35710–35718 (2019)
17. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. IEEE Trans. Sofw. Eng. 33(1), 2–13 (2007)
18. Jiang, L., Zhang, H., Cai, Z.: A novel Bayes model: hidden naïve Bayes. IEEE Trans. Sofw. Knowl. Data Eng. 21(10), 1361–1371 (2009)
19. Catal, C., Sevim, U., Diri, B: Practical development of an eclipse-based software fault prediction tool using naïve Bayes algorithm. Exp. Syst. Appl. 38, 2347–2353 (2011)
20. Dejaeger, K., Verbraken, T., Baesens, B: Toward comprehensible software fault prediction models using Bayesian network classifiers. IEEE Trans. Sofw. Eng. 39(2), 237–257 (2013)
21. Jiang, L., et al.: A correlation based feature weighting filter for naïve Bayes. IEEE Trans. Knowl. Data Eng. 31(2), 201–213 (2019)
22. Cano, A., Zafra, A.S., Ventura, S. Weighted data gravitation classification for standard and imbalanced data. IEEE Trans. Cybern. 43(6), 1672–1687 (2013)
23. Turhan, B., Bener, A.: Software defect prediction: heuristics for weighted naïve Bayes. In: Proceedings of the Second International Conference on Software Data and Technology, pp. 244–249, (2007)
24. Turhan, B., Bener, A.A.: Analysis of naïve Bayes assumptions on software fault data: an empirical study. Data Knowl. Eng. 68(2), 278–290 (2009)
25. Asmono, R.T., Wahono, R.S., Syukur, A.: Absolute correlation weighted naïve Bayes for software defect prediction. J. Sofw. Eng. 1(1), 38–46 (2015)
26. Ji, H., et al.: A new weighted naïve Bayes method based on information diffusion for software defect prediction. Sofw. Qual. J. 27, 923–968 (2019)
27. Poon, W.N., et al.: Cross-project defect prediction using a credibility theory based naïve Bayes classifier. In: 17th IEEE International Conference on Software Quality, Reliability and Security, QRS 2017, pp. 434–441, Prague, (2017). https://doi.org/10.1109/QRS.2017.53
28. Malhotra, R.: A systematic review of machine learning techniques for software fault prediction. Appl. Soft Comput., 27, 504–518 (2015)
29. Hosseini, S, Turhan, B, Gunarathna, D.D: A systematic literature review and meta-analysis on cross project defect prediction. IEEE Trans. Sofw. Eng. 45(2), 111–147 (2019)
30. Hosseini, S., Turhan, B., Mantyla, M.: A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. Inf. Sofw. Technol. 95, 296–312 (2018)
31. He, P., et al.: An empirical study on software defect prediction with a simplified metric set. Inf. Sofw. Technol. 59, 170–190 (2015)
32. Zhang, F., et al.: Towards building a universal defect prediction model with rank transformed predictors. Empirical Sofw. Eng. 21(5), 2107–2145 (2016)
33. Ryu, D, Baik, J: Effective multi-objective naïve bayes learning for cross-project defect prediction. Appl Soft Comput. 49, 1062–1077 (2016)
34. Arar, O.F., Ayan, K.: A feature dependent naïve Bayes approach and its application to the software defect prediction problem. Appl. Soft Comput. 59, 197–209 (2017)
35. Jureczko, M., Madeyska, L.: Towards identifying software project clusters with regard to defect prediction. In: Proceedings of Sixth International Conference on Predictive Models in Software Engineering, p. 9, New York, September, (2010)
36. Chen, X., et al.: Software defect number prediction: Unsupervised vs supervised methods’. Inf. Sofw. Technol. 106, 161–181 (2019)
37. Lu, Z., et al.: Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction’. Autom. Sofw. Eng. 1, 1–45 (2017)
38. Herbold, S., Trautsch, A., Grabowski, J.: Comparative study to benchmark cross-project defect prediction approaches. In: Proceedings of 2018 40th IEEE/ACM International Conference on Software Engineering, pp. 1063–1063, (2018). https://doi.org/10.1145/3180155.3182542
39. Herbold, S., Trautsch, A., Grabowski, J: A Comparative study to benchmark cross-project defect prediction approaches’. IEEE Trans. Sofw. Eng. 44(9), 811–833 (2018). https://doi.org/10.1109/TSE.2017.2724538

How to cite this article: Zou Q, Lu L, Qiu S, Gu X, Cai Z. Correlation feature and instance weights transfer learning for cross project software defect prediction. IET Softw. 2021;15:55–74. https://doi.org/10.1049/sfw.2.12012