METHOD ARTICLE

An optimized approach for class imbalance problem in heterogeneous cross project defect prediction [version 1; peer review: awaiting peer review]

Lipika Goel[1], Neha Nandal[1], Sonam Gupta[2]

[1]Computer Science and Engineering, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Telangana, 500090, India
[2]Computer Science and Engineering, Ajay Kumar Garg Engineering College, Ghaziabad, Uttar Pradesh, 201009, India

v1 **First published:** 16 Sep 2022, **11**:1060
https://doi.org/10.12688/f1000research.123616.1

Latest published: 16 Sep 2022, **11**:1060
https://doi.org/10.12688/f1000research.123616.1

Open Peer Review

Approval Status *AWAITING PEER REVIEW*

Any reports and responses or comments on the article can be found at the end of the article.

Abstract

Background: In recent studies, Cross Project Defect Prediction (CPDP) has proven to be feasible in software defect prediction. When both the source as well as the target projects have the same metric sets, it is termed as a homogeneous CPDP. Current CPDP strategies are difficult to implement through projects with a variety of different metric sets. Aside from that, training data often has a problem with class imbalance. The number of defective/bug-ridden and non-defective/clean instances of the source class is usually unbalanced. To address this issue, we propose a heterogeneous cross-project defect prediction framework that can predict defects across projects with different metric sets.
Methods: To construct a prediction framework between projects with heterogeneous metric sets, our heterogeneous cross project defect prediction approach uses metric selection, metric matching, class imbalance (CIB) learning followed by ensemble modelling. For our study, we have considered six open-source object-oriented projects.
Results: The proposed model resolved the class imbalance issue and records the highest recall value of 7.5 with f-score value as 7.4 in comparison with other baseline models. The highest AUC (area under curve) value of 0.86 has also been recorded. K fold cross validation was performed to evaluate the training accuracy of the model. The proposed optimized model was validated using the Wilcoxon signed rank test (WSR) with a significance level of 5% (i.e., P-value=0.05).
Conclusions: Our empirical research on these six projects shows that predictions based on our methodology outperform or are statistically comparable to Within-Project Defect Prediction (WPDP) and other heterogeneous CPDP baseline models.

Keywords
Cross Project Defect Prediction (CPDP), Class Imbalance (CIB), Ensemble Modeling, Heterogeneous, Metric Matching

This article is included in the Software and Hardware Engineering gateway.

This article is included in the Python collection.

This article is included in the Computing Science collection.

This article is included in the Computational Modelling and Numerical Aspects in Engineering collection.

Corresponding author: Lipika Goel (lipika1670@grietcollege.com)

Author roles: Goel L: Methodology, Writing – Original Draft Preparation; **Nandal N**: Software, Writing – Review & Editing; **Gupta S**: Supervision, Visualization

Competing interests: No competing interests were disclosed.

Grant information: The author(s) declared that no grants were involved in supporting this work.

How to cite this article: Goel L, Nandal N and Gupta S. **An optimized approach for class imbalance problem in heterogeneous cross project defect prediction [version 1; peer review: awaiting peer review]** F1000Research 2022, **11**:1060 https://doi.org/10.12688/f1000research.123616.1

First published: 16 Sep 2022, **11**:1060 https://doi.org/10.12688/f1000research.123616.1

Introduction

Every Software Quality Model (SQM) mainly includes two operations for achievement of a good quality software; the first is the software quality assurance (SQA) for achieving the best quality and the second is the software defects prediction (SDP) for making predictions for maximum defects. This ensures the best quality software product is used for the prediction. Many SDP models have been built by using various data mining techniques over the last few decades that use the software defect databases.[1]

Within-Project Defect Prediction (WPDP)

Defect prediction models are mainly designed to work within a project defect prediction. In such a scenario, the defect prediction model is trained using a partial dataset (i.e., having defective or non-defective labels) of a project (the training set) and is tested for the remaining dataset (for which the labels are predicted) of the same project (the testing set).[2]

Cross-Project Defect Prediction (CPDP)

However, in current times, with increased technologies and demands for applied technologies, the challenge for defect prediction also increases manifold as it is not a cost-effective process to find the labeled dataset for all projects to be applied for a within-project defect detection model training.[3] When a software is newly built and there is no historical record of defects of that project, how can the defects be predicted for such a newly-developed project?[4]

Approaches have been made for CPDP.[5] In such approaches, a prediction model is trained using the dataset instances of one project (training project) that are labeled instances and are tested for unlabeled instances of another project (the testing project). CPDP can be further categorized into the following: 1) The model that is trained using only common metrics that exist in both training as well as in the testing project is called the Homogeneous CPDP (HDP). In such a technique, the predictions are made for unlabeled instances of the testing project. When the metrics/features are specific to the training and the testing project, i.e., the source and the target projects have different feature sets, then it becomes a challenge for the prediction model to predict the defects of the target project. This problem may arise due to project variations written in different languages[6] and leads to poor accuracy for defect prediction.[7] This is called the Heterogeneous CPDP (HCPDP). Figure 1 gives the description of WPDP, HDP and HCPDP.

In the classification problem of HCPDP, the target class has a binary subclass (classes 0 and 1), i.e., defective and non-defective. The total number of instances in these two subclasses are mostly not identical. A class imbalance (CIB) problem occurs when there is a distributional disparity between the defective and non-defective classes, resulting in an unbalanced training dataset. When the distribution of the two classes is imbalanced, it leads to a biased prediction.

In this paper, an optimized approach to handle the CIB issue in HCPDP is proposed. First, the key idea of this framework is matching the metrics of the source and target datasets. It will then deal with the dataset's disparity by partitioning the training data into data frames with roughly equal numbers of non-defect and defect susceptible groups in each data frame. Second, this proposed framework will also perform HCPDP. Maximum voting is used in the ensemble model. The training accuracy of the proposed framework is assessed using K fold cross validation. Finally, the proposed framework was validated using the Wilcoxon signed rank test.[32] This research report addresses the following research questions:

- Q1. Are the results obtained using optimized approach comparable to WPDP?
- Q2. (i) Does the proposed model resolve the CIB problem in the source data?

  (ii) Does it outperform the HCPDP model having imbalanced dataset?

The significant contributions are:

- To propose an optimized approach for the CIB issue in HCPDP.
- To develop a standalone ensemble framework for HCPDP.

The following is the paper's classification: The HCPDP literature review is discussed in Section 2. Preliminaries illustrate the datasets, feature selection process, and correlation methodology in Section 3. The proposed ensemble framework, which involves the flow from data acquisition to modelling, is highlighted in Section 4. The experimental setup is outlined in Section 5, and the experimentally observed results are highlighted in Section 6. The validation of the result is covered in Section 7. Section 8 brings the paper to a close.

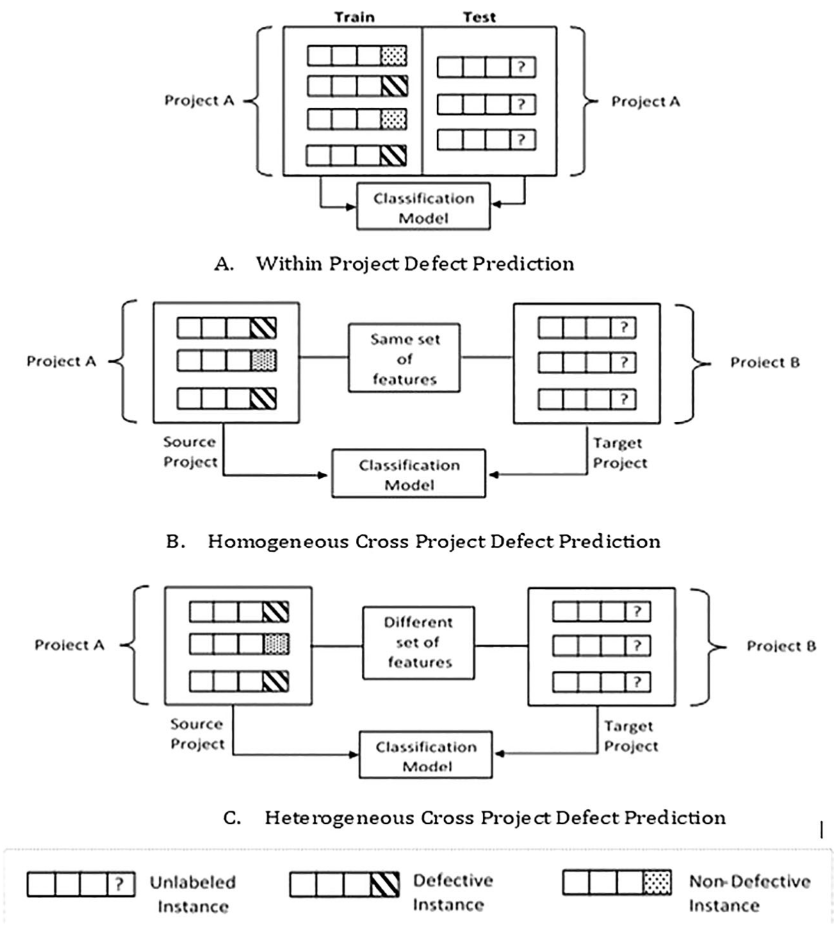


Figure 1. Description of WPDP (Within-Project Defect Prediction), HDP (Homogeneous Cross Project Defect Prediction), and HCPDP (Heterogeneous Cross Project Defect Prediction).

State of art

In 2002, researchers presented their work on MARS- Multivariate Adaptive Regression spline which got attention in CPDP.[8] The work done was on data design of Xpose and Jwriter.[8] The model has been trained using Xpose dataset and prediction was done in Jwriter for fault prediction. This work has been compared with Regression on Linear basis in which MARS model provided better results.

In the year 2009, researchers utilized different discrete sources for the defect datasets of the different software and pre-processed the data for prediction by removing redundant, noisy and irrelevant data for model training. The work has been performed on 10 different projects by using Nearest Neighbor methodology.[5] The conclusion is that the model performed well for WPDP. Work on transformation of log to find similarity in testing and training projects to avoid project dependency was also done in the year 2009.[7] In the year 2011, investigators worked on search engines for defect prediction. Classification has been performed using process parameters and coding standards. The training of the model has been done with the search engine of Mozilla Firefox and prediction was performed on Internet Explorer. The results presented claimed that proposed model performs well with Mozilla Firefox as testing project[6] Also it has been asserted by the authors in this work, that the quality and the accuracy of the defect model can vary when examined with different aspects. The work is being proved with the experiments.[6] The results states that local behavior is better in comparison of global one. In the year 2012, a scholar proposed experimental work by considering evaluation measures including precision, recall and F-measure. The presented work states that the mentioned evaluation measures are not enough to give quality assurance for defect prediction with different models. It has been declared that area under the curve provides efficient measurement in WPDP.[9]

In the year 2013, authors launched a multi-objective approach for overcoming the problems of single objective model. The Logistic Regression model was being trained in this work with NSGA-II (non- dominated sorted Generic Algorithm).[10] In 2014 researchers presented a Universal Defect Prediction Model in which total 1398 projects being utilized from source forge and Google code. The matrices were being compared between test and training projects. The mat of at least 26 matrices was being considered a success and predictions made.[11] Further in this study, authors considered a new metric as characteristic vectors of instances to overcome the upper limitation.[12] The comparison of feature disparity was being done with CPDP and the results found were negative. This experiment was done with 11 projects using three datasets. They also worked on with-in project defect prediction-WPDP and cross project defect prediction-CPDP with feature selection methodology for comparison of performance. It has been observed that higher precision was achieved in WPDP with less training project features. Also, CPDP provided better results of Recall and F-score.

In the year 2015, investigators presented the work on CCA- Canonical Correlation Analysis for defect prediction. It was the first work to present the concept of Heterogeneous Defect Prediction. The metrices disparity problem was being resolved in this work by embedding dummy metrices with null values. The experiment was done with 14 projects using 4 datasets.[13] Further scholars utilized a novel approach of transfer cost-sensitive boosting methodology which provided CPDP results.[14] Also, an approach of CPDP with multi objective Naïve Bayes technique was introduced with CIB which performed better than all other models of WPDP along with the single objective models. In further study of HCPDP researchers launched HDP- Heterogeneous Defect Prediction concept.[15] An optimized HDP with metric matching and selection was also proposed.[16] The work has been done on 28 different projects and the results were compared with WPDP which performed better than some statistical projects.

In the year 2016, authors worked on transfer learning method having 34 projects with five datasets. The metrices with null values were not being embedding as in CCA. The results of the work were being compared with WPDP.[17]

Comparison on filtration methods for defect prediction was done in 2017.[18] It has been stated in the work that choosing right filtration method can highly impact the model's capability. The four methods of filtration (Data Characteristic based Filter-DCBF, Source project data Guided Filter -SGF, Target project data Guided Filter -TGF and Local Cluster based Filter-LCBF) were being compared in this work. HSBF- Hierarchical Selection Based Filter has been proposed in this work to overcome the cons of existing filters in context of large datasets for enhancing scalability.

Further in 2017 a novel approach of FESCH (Feature Selection using Clusters of Hybrid)-data was proposed which performed better than ALL, WPDAP and TCA+ in different domains. The results achieved were independent of the classifiers utilized and very sustained.[19] In 2018, authors worked on reducing the higher dimension features by using domain adaptation technique. The Dictionary learning techniques has been applied to understand the feature space differences. Three open source projects including NetGene, AEEEM, and Nasa have been utilized with the evaluation measures of F-measure and Recall for comparison of HAD- Heterogeneous Defect Adaptation.[20] Further investigation and comparison of existing models of CPDP was done and authors searched for the best suited methods. AUC (area under curve) was being utilized for performing comparison with other works to check the difference in results. A simple Neural Network was being proposed for CPDP in the year 2019 to tackle HDP in which cross entropy function was applied for classification of error.[21]

In the year 2021, researchers presented the work on semi-supervised learning for tackling heterogeneous defect prediction. The open-source projects were being utilized for the analysis. The metric representation and canonical correlation analysis has been introduced to analyze different company projects.[22] A total of 26,407 modules from GitHub has been collected which was unlabeled dataset and can be extended as per the requirement. Results declared stated that prediction has been optimized through this work. They presented a work for training and prediction by using different companies for which cross project domain prediction task was utilized. The conclusion of the work states that cost of FPR was increasing as DDR (defect detection rate) was increasing. Further in 2021, authors have utilized correlation coefficients for heterogeneous defect prediction.[23] The work has been done completely experimental and it has been compared with the baseline models. Results declared stated that the proposed model performed better than other base models.

In the year 2022, investigators worked on the concept of heterogeneous feature selection by utilizing nested stacking.[24] The work presented the experiments done on two datasets i.e. KAMEI and PROMISE. The demonstration of the work has been done through two evaluation indicators of Area under the curve and F1-Score. Results presented declared that the proposed model outperformed other baseline models.

The major gap identified in the state of art is the solution to the class imbalance issue in HCPDP. This issue arises due to imbalance in number of instances in faulty and non-faulty situation. This issue obstructs the effectiveness of defect prediction models in practical life scenario. Although many studies have been performed considering this issue, yet there is the need to take this issue into more consideration for more optimized solution to the CIB problem mainly in HCPDP.

Table 1. Details of the datasets.

S.No.	Group	Dataset	Total No of instances	No. of defective instances	No. of non-defective instances
1	NASA	Pc2	705	79	626
2		Pc3	1077	134	943
3		Pc4	1458	178	1280
4	SOFTLAB	Ar3	63	8	55
5		Ar4	107	20	87
6		Ar6	101	15	86

Preliminaries

Datasets

This HCPDP study relied on well-defined datasets. For the experiment, six open-source projects were considered, and the datasets were taken from the OpenML (an open platform for sharing datasets. The various versions of these datasets can be downloaded for free. Please see *Underlying data*[34] for information on where the full data can be accessed).

Data related to each class of the project was taken for the experiment. Table 1 contains a summary of the dataset. We considered the object-oriented projects of NASA and SOFTLAB with different sets of features in our analysis since we concentrated on HCPDP.

Feature selection

The Extra Tree Classifier[33] is one of the ensemble learning techniques in which the output produced by several distinct decision trees, which are not mutually correlated, aggregated as a “forest,” are combined to generate the predictive model. For feature selection, the Gini Index also known as Gini Information of each feature is computed. To perform feature selection, the features are sorted in a descending order of their Gini Importance. The user then selects the top k features as per the experiment.

- STEP 1: Firstly, build an extra tree forest over the given data set with as many decision trees of choice. For each decision tree we would select the no. of attributes/features in the random sample;
- STEP 2: In construction of the forest, for every attribute/feature, the normalized total reduction which is used to split the feature in the decision tree is computed (Gini Importance of the feature).
- STEP 3: Now, the features are ranked based upon their Gini Importance value and one can select the top features of his/her choice.

The following approach is used in our work to select the k important attributes/features from the target and the source projects.

For calculating the Gini Importance, the first entropy is calculated as per the following formula:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i) \tag{1}$$

where, c is the number of unique class labels and p_i is the proportion of rows with output label as i.

$$Gain(S,A) = Entropy(S) - \sum_{veValues(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

where, “A” represents the feature/metrics.

Spearman correlation

To understand the Spearman correlation,[11] we need to infer monotonic function. A monotonic function is one that is completely non-increasing or non-decreasing. A non-parametric statistical test called Spearman correlation ranks the intensity of a monotonic relationship between two variables. The coefficient of correlation is often designated by ρ or rS which returns a value from -1 to 1. The test does not contain any presumption regarding the distribution of data and returns the correlation analysis for the least ordinal measured variables.

The Spearman formula to evaluate rank correlation is as follows:

If there are no tied ranks, then we use the following formula:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{3}$$

where, d_i = difference in paired ranks and n = number of cases.

The formula when there are tied ranks is:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{4}$$

where, i = paired score.

In our experiment, we determined the correlation between the training and the testing metrics of the important features. In Section 4, details of metrics matching are stated.

Class imbalance

Prediction of the target class for a given instance of data is the main objective of classification predictive modeling. When the distribution of the two classes is imbalanced, it leads to biased prediction. The distribution difference in the minority and the majority class leads to the imbalanced classification and is known as CIB problem.[25] Most of the machine learning models are designed with an assumption of equal number of instances in each class. The imbalanced nature of the dataset results in poor predictive performance of the model. Spam prediction, defect prediction, churn prediction are some of the real-world classification problems with class imbalance.[26] Sampling by oversampling of the minority instances or undersampling of the majority class is one of the simplest way to solve the imbalance nature of the classification problems. Oversampling by introducing the duplicate instances may lead to the problem of overfitting. Undersampling may sometimes lead to loss of important information. Synthetic Minority Oversampling Technique (SMOTE) and Random Undersampling (RUS) are few algorithms that are oversampling and undersampling techniques.[27] Figure 2 depicts an imbalance distribution of the two classes.

Ensemble learning model (random forest & XGBoost)

Machine learning withholds ample of classifiers. We need to classify observations accurately in order to achieve the required outcome. Random forest classifier is one of the prime classifiers used to convert the unreliable data model like the decision trees to make a more robust model.[28] Random forest's building component is the decision tree, which is a spontaneous model. It's an illustrative model achieved through the questionnaire framed about the data until it reaches a decisive point. The random forest is a collection of such trees, each of which reveals the class prediction, with the tree with the most votes becoming the model's forecast.

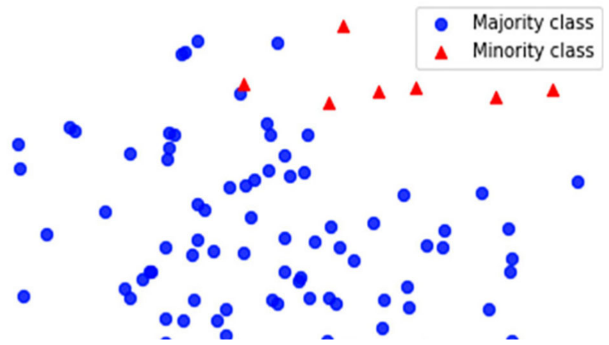


Figure 2. Imbalance distribution of two classes. Authors' own figure.

Random forest can be converted into the robust model through bagging Random Forest takes benefit of this by having each tree to sample from the data set at random with replacement, resulting in unique trees.[29] This is termed as bagging. Because of Bagging variance is reduced through Boosting. It reduces bias by instructing the consecutive model by telling it what errors the preceding models made (the boosting part).

The two major algorithms used for boosting are:

- Adaptive boosting: It is basically an algorithm that converts the weaker classifier into the stronger one.
- Gradient boosting: It is a method of training each consecutive model based on the past outcome.

XGBoost is the technique of boosting the decision tree gradiently, which turns out to be highly flexible and efficient.[30] It is an approach for high performance and speed. It's the ability of parallel computation on a single machine that makes this algorithm faster. What it does is in spite of considering the loss for all possible splits to generate a new branch, it considers the entire distribution of the features over the data set and uses this information to drastically improve on the search space accountable for feature split.

Proposed methodology

The flowchart of the proposed framework is presented in Figure 3. The proposed methodology is discussed under the following heads:

1. Data Acquisition and Understanding.
2. Data Preprocessing and Preparation.
3. Modeling.

Data acquisition and understanding

In this study, we have considered six open sources object-oriented projects. The details of the projects are mentioned in Table 1. Various combinations are made to form source–target project pairs from existing projects. The feature sets of all the projects considered are not similar. The datasets used are described in detail in Section 3.

Data pre-processing and preparation

Data pre-processing is the most effective way of improving the model's performance as it enhances the quality of both training and testing data. First, the data are analyzed for missing and null values. We have used the average method to fill

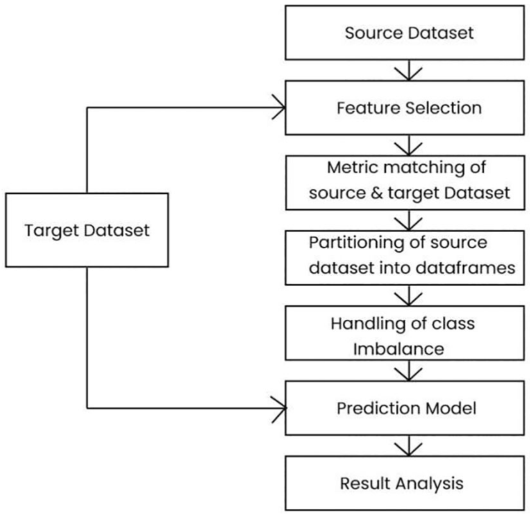

Source Dataset
Feature Selection
Metric matching of source & target Dataset
Target Dataset
Partitioning of source dataset into dataframes
Handling of class Imbalance
Prediction Model
Result Analysis

Figure 3. Flowchart of the proposed framework.

Table 2. **The list of important features from each open-source project considered in this study.**

PC2	PC3	PC4	AR3	AR4	AR6
Halstead Volume	Decision Density	Loc_Code_And_Comment	Total_Operators	Halstead Volume	Halstead Length
Num Operands	Number of Lines	Percent_Comments	Halstead_Length	Halstead Length	Executable Loc
Halstead Length	Call Pairs	Loc_Comments	Halstead_Time	Halstead Difficulty	Halstead Error
Halstead Prog Time	No. of Unique Operands	Loc_Blank	Halstead_Effort	Halstead Effort	Condition Count
Num Unique Operators	Parameter Count	Num_Unique_Operators	Cyclomatic_Complexity	Halstead Error	Branch Count
Loc Comments	Halstead Content	Loc_Total	Halstead_Vocabulary	Total Operands	Decision Count
Halstead Difficulty	Loc Code & Comment	Halstead_Length	Design_Density	Unique Operands	Halstead Level
Percent Comments	Loc Comments	Cyclomatic_Density	Halstead_Error	Total Loc	Cyclomatic Density
Halstead Effort	Percent Comments	Node_Count	Branch_Count	Halstead Vocabulary	Normalized Cyclomatic Complexity
Parameter Count	Loc Blank	Call_Pairs	Total_Operands	Unique Operators	Cyclomatic Complexity

in the missing values for any particular attribute. For encoding purposes, we have used a label-encoder. This is followed by the data normalization process. The pre-processing was carried out using Python 3.8.3.

Modeling

A set of classifiers are combined in an ensemble learning model, generated in Python 3.8.3 (see *Software availability*[35]). All experiments in this study are performed on Python 3.8.3. This increases the classification model's overall efficiency. When all of the base classifiers are integrated, a data tuple is chosen. The class label conclusion is made based on the popular vote of all the basic classifiers. The following are the phases of modelling in the proposed framework.

Feature and metric selection

For each of the training and testing datasets, the Extra Tree Classifier implemented in Python 3.8.3 (see *software availability*[35]) has been used for selection of important features. The working of the Extra Tree Classifier has been discussed Section 3.[31] The top 15% of metrics were selected using feature selection technique. Top ten features have been selected with the highest Gini Importance.[31] The list of features selected from each dataset is stated in Table 2. Figure 4 is a graph generated for depicting the Gini Importance of top ten features of each of the considered projects in defect prediction.

Matching metrics

The source and target datasets have different features/metrics. The similarity between each set of source and target features/metrics is examined using the Spearman's correlation technique. The essential concept is to calculate matching/correlation scores for all combinations of source/training and target/testing data. Figure 5 depicts a sample matching.

Let us consider two sources, training/metrics (A1, A2) and target/testing metrics (B1, B2). Thus, there are four possible combinations of matching pair of metrics, i.e., (A1, B1), (A2, B1), (A1, B2) and (A2, B2). A specific threshold value of cutoff is set to discard the poorly correlated/matched metric pairs. In our experiment the threshold cutoff value is set to

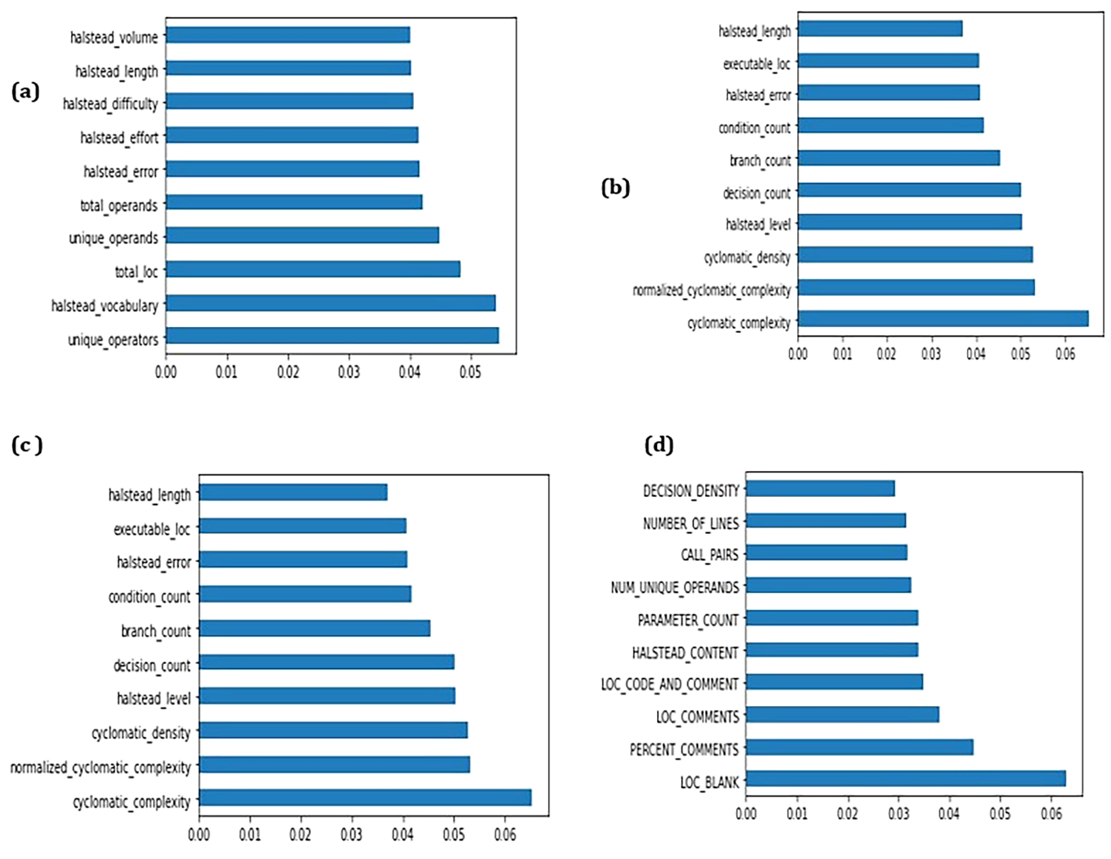


Figure 4. Gini Importance of features of projects ar4, ar6, pc2, pc3 in a, b, c, d respectively (Y axis: Features and X axis: Gini Importance values).

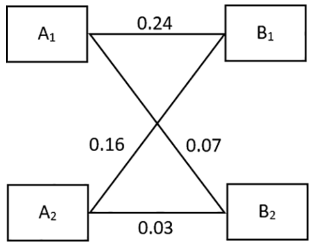


Figure 5. Sample matching.

0.05 since it is commonly used these days due to its positive impact on predictions.[11] Now, we include only those pairs whose matching score is >0.05 and build the ensemble prediction model with a matching score >0.05. In Figure 5, the correlation between (A2, B2) is discarded since the correlation/matching value is <0.05. Thus, the matching pairs to be considered include (A2, B1), (A1, B1) and (A1, B2). Therefore, in order to convert the relationship to one-to-one, we selected the ones with the maximum matching scores. Besides, the main idea was also to have a maximum no. of selected attributes/features in the source and target pair. Table 3 tabulates a few samples of the final selected source and target matching pairs with their matching scores.

Class imbalance learning

The datasets used in this study are extremely unbalanced. There is a considerable difference in the overall ratio of defect-prone and non-defect-prone categories. This will lower the model's overall output and is known as the CIB learning issue. The no. of defective and non-defective groups for projects can be found in Table 1. The following steps, as shown in Figure 6, are taken to overcome this issue.

The number of defect prone and non-defect prone groups in the source/training dataset after selecting important and correlated features in the source and target projects are separated.

Let x represent the no. of defect-prone cases and y represent the number of defect-free instances. The number of non-defective classes is almost alpha (=7 in the experiment) times the number of defective classes, i.e. x/y=α.

As a result, the training dataset's non-defect prone occurrences were partitioned into frames ((1… y, y+1…2y, 2y+1…..3y, 3y+1…..4y… (-1)y+1….y). ince the value of =7, the non-defective instances of the training dataset were segmented into 7 frames in this experiment. The training dataset's defect-prone occurrences are attached to each data frame after separation. The training dataset's defect-prone samples are appended to each data frame after segregation.

As a result, in each frame, the faulty instances, i.e. y, are added to x1…x, resulting in an approximately equivalent number of defect prone and non-defect prone occurrences. The ensemble model is now fed with these seven independent data frames, which have been balanced. Figure 7 depicts the ensemble approach of proposed methodology.

Building the prediction model

The modelling includes a model fitting and voting method. There are seven alternative balanced dataframes after resolving the CIB problem for each training dataset.

Random Forest and XGBoost are ensemble models and improved the performance of the classification models; therefore, in the experiment the odd-numbered dataframes have been modeled using the Random Forest classifier and the even-numbered dataframes have been modeled using the XGBoost.

For a specific instance, the outputs of all these classifiers are considered and have been ensembled further using maximum voting. Figure 7 gives the ensemble approach for proposed methodology.

Model fitting

For the seven main dataframes, Random Forest and XGBoost are used as classifiers. To boost performance, the learning model's parameters are set to a certain numeric value. Hyperparameter tuning is the term for this technique. The learning model is hyperparameter optimized in this experiment. These parameters must be precisely specified rather than inferred from data.

Table 3. (i), (ii), (iii) - Source and target matching pairs with their matching scores.

(i) (PC2→AR4)										
AR4 PC2	Halstead Volume	Halstead Length	Halstead Difficulty	Halstead Effort	Halstead Error	Total Operands	Unique Operands	Total Loc	Halstead Vocabulary	Unique Operators
Halstead Volume	0.25									
Num Operands										0.18
Halstead Length		0.22								
Halstead Prog Time					0.19					
Num Unique Operators							0.17			
Loc Comments								0.18		
Halstead Difficulty			0.16							
Percent Comments									0.21	
Halstead Effort				0.18						
Parameter Count						0.21				

(ii) (PC3→AR3)										
AR3 PC3	Total_Operators	Halstead_Length	Halstead_Time	Halstead_Effort	Cyclomatic_Complexity	Halstead_Vocabulary	Design_Density	Halstead_Error	Branch_Count	Total_Operands
Decision Density							0.16			
Number of Lines									0.21	
Call Pairs		0.18								
No. of Unique Operands	0.23									
Parameter Count										0.18
Halstead Content				0.19						
LOC Code & Comment					0.23					
LOC Comments								0.16		
Percent Comments						0.21				
LOC Blank			0.18							

Table 3. *Continued*

(iii) (PC3→AR6)										
AR6 PC3	Halstead Length	Executable Loc	Halstead Error	Condition Count	Branch Count	Decision Count	Halstead Level	Cyclomatic Density	Normalized Cyclomatic Complexity	Cyclomatic Complexity
Decision Density						0.16				
Number of Lines					0.18					
Call Pairs	0.19									
No. of Unique Operands		0.18								
Parameter Count								0.18		
Halstead Content			0.19							
LOC Code & Comment				0.21						
LOC Comments									0.22	
Percent Comments										0.15
LOC Blank							0.17			

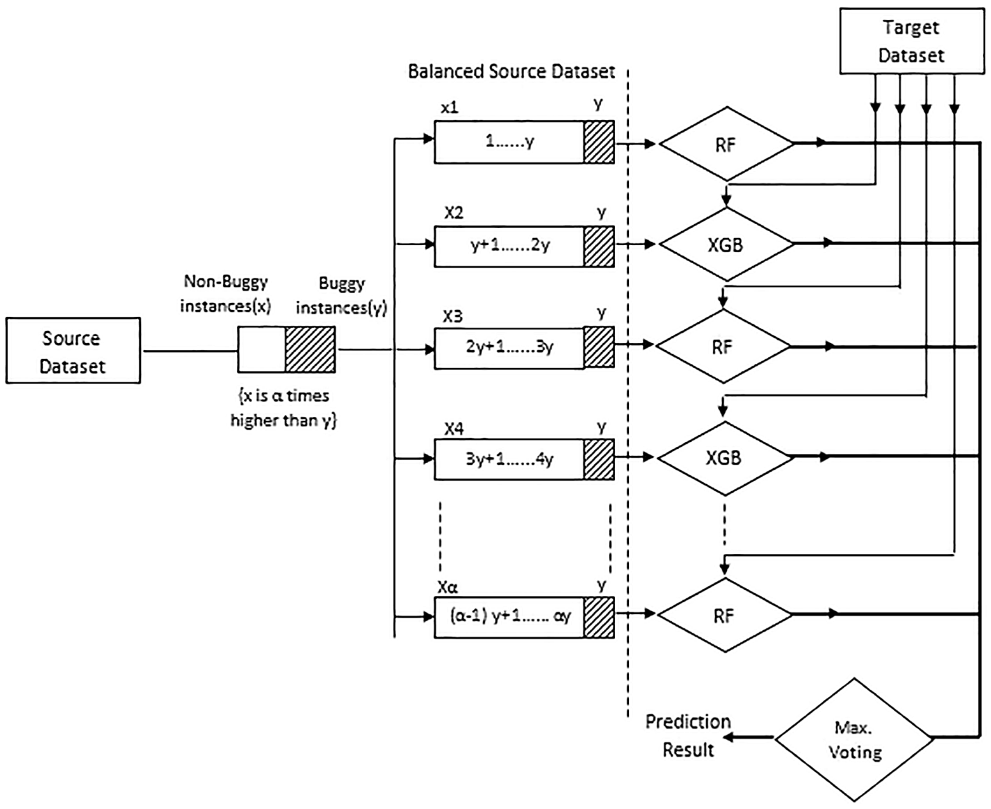


Figure 6. **Optimized approach for resolving class imbalance.** XGB, XGBoost; RF, Random Forest.

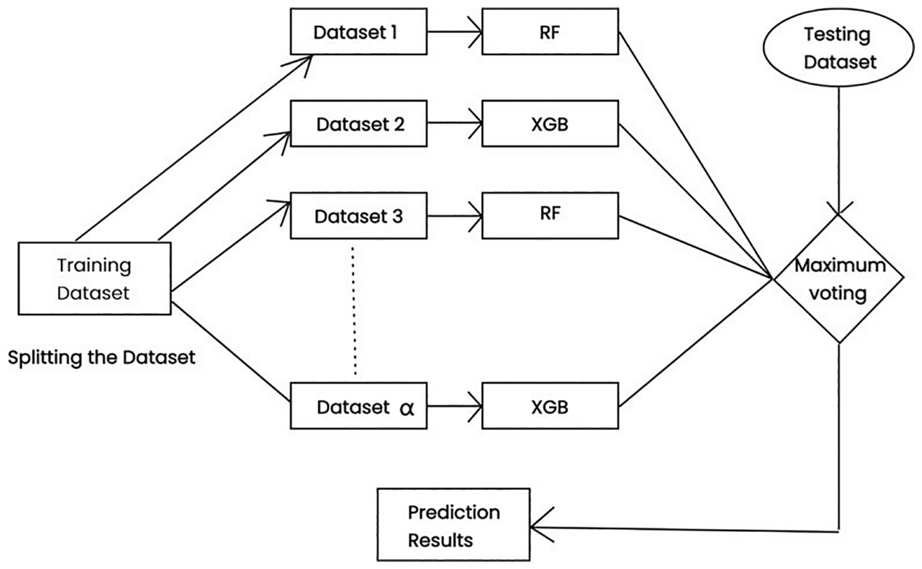


Figure 7. **Ensemble approach of the proposed methodology.** XGB, XGBoost; RF, Random Forest.

Voting system

Maximum voting has been used to develop a new ensemble model. Each model's prediction in relation to the data frames is taken into account. The class with the most votes is chosen as the binary classification's final prediction for any given case.

Experimental setup

Research questions

To systematically evaluate the proposed HCPDP models, two research questions are set:

- Q1. Is the proposed model comparable to WPDP?
- Q2. (i) Is the proposed model capable of addressing the source dataset's CIB issue?

  (ii) Does it outperform the HCPDP model with the imbalanced dataset?
- Q1 and Q2 lead us to investigate whether our HCPDP model is comparable to WPDP (Baseline1) and CDDP with imbalanced dataset (Baseline2).

Baselines

The work in this paper has been compared with two baselines: WPDP (Baseline 1) and HCPDP with imbalance source data (class imbalance). Comparing the novel framework with within project defect prediction gives statistical evidence of the feasibility and applicability of our model. The proposed framework handles the CIB problem and predicts the target class, and hence we have included HCPDP with CIB also as baseline (Baseline 2).

Experimental design

Three experiments were conducted to evaluate the applicability and predictive performance of the framework. For the machine learning classification model, we have used Random Forest and XGBoost.

Experiment 1: For WPDP, the source dataset was divided in the ratio of 70:30 (randomly) as training and testing, respectively. The classification model was applied, and the results were noted.

Experiment 2: After application of a metric selection and matching, the imbalanced source dataset was used for training the classification model. The prediction of the target instance of the testing dataset was noted.

Experiment 3: The proposed m framework (as discussed in Section 4) was used for training and testing of the source and target project pair, respectively. The prediction results were noted and analyzed.

Experimental results and discussion

This segment provides a description of the findings from the previous experiments. Tables 4, 5, and 6 summarizes the findings. The Wilcoxon Signed Rank (WSR) test is used to further validate the proposed framework. The values of the

Table 4. Accuracy, recall, F-Score and AUC (area under curve) using the proposed framework.

Dataset		Accuracy		Recall		F-Score		AUC
Training	Testing	Training	Testing	True	False	True	False	
PC2	AR3	0.81	0.75	0.81	0.75	0.87	0.62	0.775
	AR4	0.83	0.72	0.79	0.72	0.78	0.75	0.792
	AR6	0.89	0.78	0.83	0.81	0.81	0.83	0.812
	Avg.	0.843	0.750	0.810	0.760	0.820	0.733	0.793
PC3	AR3	0.84	0.7	0.76	0.71	0.73	0.75	0.679
	AR4	0.86	0.69	0.71	0.7	0.74	0.72	0.778
	AR6	0.78	0.74	0.78	0.75	0.69	0.73	0.746
	Avg.	0.827	0.710	0.750	0.720	0.720	0.733	0.734
PC4	AR3	0.84	0.81	0.82	0.81	0.79	0.74	0.894
	AR4	0.86	0.75	0.78	0.76	0.78	0.76	0.789
	AR6	0.82	0.68	0.78	0.74	0.74	0.76	0.784
	Avg.	0.840	0.747	0.793	0.770	0.770	0.753	0.822

Table 5. **Accuracy, recall, F-Score and AUC (area under curve) using HCPDP (Heterogeneous Cross Project Defect Prediction) with class imbalance.**

Dataset		Accuracy		Recall		F-Score		AUC
Training	**Testing**	**Training**	**Testing**	**True**	**False**	**True**	**False**	
PC2	AR3	0.82	0.71	0.79	0.71	0.82	0.63	0.772
	AR4	0.83	0.74	0.75	0.69	0.77	0.71	0.612
	AR6	0.76	0.77	0.76	0.76	0.74	0.83	0.685
	Avg.	0.803	0.740	0.767	0.720	0.777	0.723	0.690
PC3	AR3	0.72	0.73	0.75	0.69	0.68	0.72	0.621
	AR4	0.75	0.65	0.72	0.67	0.74	0.69	0.812
	AR6	0.79	0.71	0.76	0.72	0.7	0.68	0.712
	Avg.	0.753	0.697	0.743	0.693	0.707	0.697	0.715
PC4	AR3	0.78	0.75	0.83	0.74	0.75	0.75	0.774
	AR4	0.76	0.78	0.79	0.73	0.74	0.77	0.796
	AR6	0.81	0.62	0.74	0.75	0.73	0.73	0.785
	Avg.	0.783	0.717	0.787	0.740	0.740	0.750	0.785

Table 6. **Accuracy, recall, F-Score and AUC (area under curve) using WPDP (Within-Project Defect Prediction).**

Dataset		Accuracy		Recall		F-Score		AUC
Training	**Testing**	**Training**	**Testing**	**True**	**False**	**True**	**False**	
PC2	AR3	0.91	0.85	0.75	0.78	0.82	0.67	0.774
	AR4	0.92	0.86	0.81	0.74	0.83	0.78	0.784
	AR6	0.88	0.89	0.79	0.76	0.78	0.84	0.878
	Avg.	0.903	0.867	0.783	0.760	0.810	0.763	0.812
PC3	AR3	0.94	0.84	0.75	0.72	0.75	0.77	0.791
	AR4	0.83	0.89	0.82	0.76	0.81	0.83	0.765
	AR6	0.88	0.84	0.83	0.81	0.79	0.79	0.778
	Avg.	0.883	0.857	0.800	0.763	0.783	0.797	0.778
PC4	AR3	0.91	0.84	0.76	0.72	0.78	0.78	0.889
	AR4	0.93	0.86	0.77	0.75	0.75	0.77	0.852
	AR6	0.89	0.89	0.82	0.81	0.81	0.79	0.863
	Avg.	0.910	0.863	0.783	0.760	0.780	0.780	0.868

performance metrics are tabulated in Table 4 using the novel/proposed method. Table 5 shows the results of using HCPDP with an imbalanced source dataset, while Table 6 shows the results of using WPDP.

In this experiment, we used a 7-fold cross validation method. The value of k = 7 was chosen so that the source dataset's training and testing partitions are broad enough to statistically represent it. However, no standard protocol for determining the value of k has been developed. Figure 8 depicts the proposed framework's 7-fold cross validation.

The two research questions as stated in the previous section are analyzed here. The AUC value determines the performance of the model. In defect prediction, the true value of Recall determines the number of defective classes in the project, which were correctly identified as defective. Therefore, this is an important factor in CPDP as identification and testing of the defective class is more vital than identification of the non-defective classes. Owing to the above reason, more focus on the true value of Recall and AUC has been referred to answer the research questions.

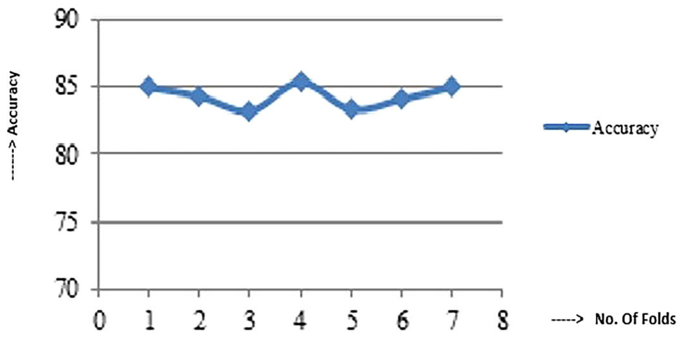


Figure 8. 7-fold cross validation for the proposed framework.

Q1. Is the proposed model comparable to WPDP?

Ans. From Table 4 and Table 6 the following inferences were made:

Using the WPDP approach, the average true values of Recall of PC2, PC3 and PC4 as source project are 0.783, 0.800, and 0.783, respectively; whereas, using the proposed framework these values are 0.810, 0.750 and 0.793, respectively. Five out of nine (i.e., 55.55%) source target project combinations using the proposed framework have a true Recall value >WPDP. As a result, it produces statistically significant findings that are superior or equivalent to WPDP.

Similarly, using the WPDP approach, the average true value of AUC of PC2, PC3 and PC4 as source project are 0.812, 0.778, and 0.868, respectively; whereas, using the proposed framework these values are 0.793, 0.734, and 0.822; respectively. Four out of nine (i.e., 44.44%) source target project combination using proposed framework have an AUC value >WPDP. As a result, it produces statistically significant findings that are superior or equivalent to WPDP. From the above two observations, it can be inferred that the proposed framework has comparable performance to WPDP and is feasible.

Q2.i. Does the proposed model handle the CIB problem of source dataset?

ii. Does it outperform the HCPDP model with the imbalanced dataset?

Ans: From Table 4 and Table 5 the following inferences were made:

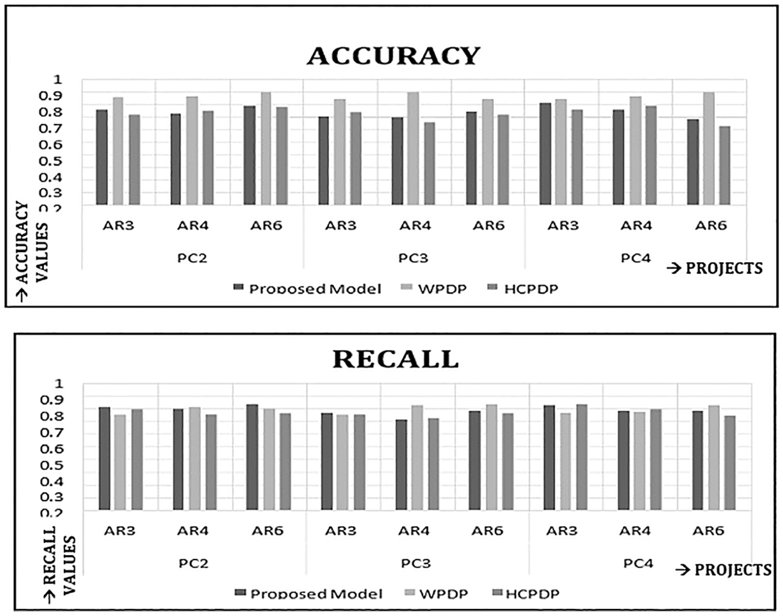
Using the HCPDP approach with the imbalanced source dataset, the average true value of Recall of PC2, PC3 and PC4 as source project are 0.767, 0.743, and 0.787, respectively; whereas, using the proposed framework these values are 0.810, 0.750 and 0.793, respectively. Seven out of nine (i.e., 77.7%) source target project combination using proposed framework have true Recall value >WPDP. Hence, it leads to better results against HCPDP approach with imbalanced source dataset with statistical significance. Similarly, using the HCPDP approach with the imbalanced source dataset approach, the average true value of AUC of PC2, PC3 and PC4 as source project are 0.690, 0.715, and 0.736, respectively; whereas, using the proposed framework these values are 0.793, 0.734, and 0.822, respectively. Six out of nine(i.e., 66.6%) source target project combination using proposed framework have AUC value greater than that of the HCPDP approach with imbalanced source dataset. Hence, it leads to better results against the HCPDP approach with the imbalanced source dataset with statistical significance.

From the above two observations, it can be inferred that the proposed framework handles the CIB problem thereby leading to better results than the HCPDP model with the imbalanced dataset approach. Figure 9 plots the comparative graph of AUC, Recall (True), F-Score and Accuracy using Proposed Framework, WPDP and HDP.

Table 7 shows the comparison among the results of some of the existing models and the proposed model, in terms of average F-score and Recall values. Also, the clear view of the same comparison is illustrated in line graph shown in Figure 10.

TNB (transfer naïve bayes) model showed the least f-score but significant rise in Recall value. TCA+ showed the least Recall value. The proposed model exceeds the performance from all the existing CPDP models including the HDP task in

(a)



(b)

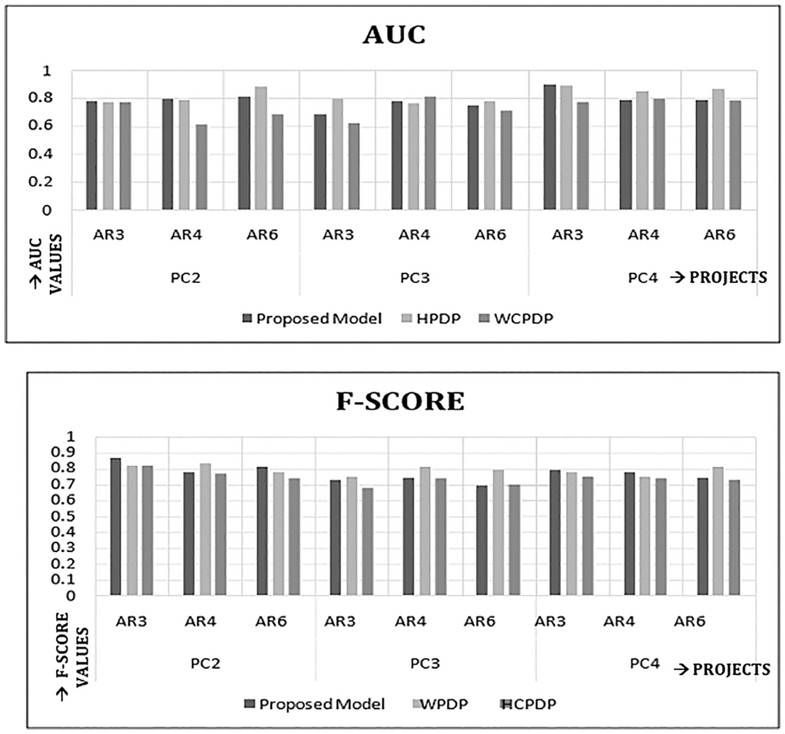


Figure 9. Comparative graph of (a) accuracy and recall (true), and (b) AUC (area under curve) and F-Score using proposed framework, WPDP (Within-Project Defect Prediction) and HCPDP (Heterogeneous Cross Project Defect Prediction).

terms of both scores having a significant surge to near about 0.75. However, the proposed model is comparable to the WPDP results with a minor difference of 0.1

Wilcoxon signed rank test

The proposed optimized model was validated using the Wilcoxon signed rank test (WSR)[32] with a significance level of 5% (i.e., P-value=0.05). This test determines if our proposed ensemble framework and the Random Forest classifier have any major differences. In order to conduct the evaluation, we took into account the Recall and AUC measures of both versions. The null hypothesis considered in this scenario is:

Table 7. Comparison of proposed framework with the existing models. CPDP, Cross Project Defect Prediction; TNB, transfer naïve bayes; CCA, Canonical Correlation Analysis; WPDP, Within-Project Defect Prediction; HCPDP, Heterogeneous Cross Project Defect Prediction.

Heterogeneous CPDP		
	AVERAGE	
	PD (recall)	**FSCORE**
TNB	0.59	0.33
NN FILTER	0.55	0.45
TCA+	0.47	0.36
CCA	0.7	0.74
PROPOSED MODEL	0.75	0.74
WPDP	0.76	0.78
HCPDP (WITH CLASS IMBALANCE)	0.75	0.72

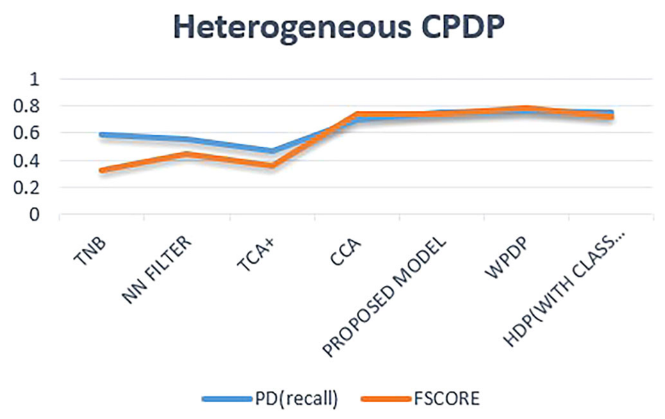


Figure 10. Graphical representation of the comparison of proposed framework with the existing baseline models. CPDP, Cross Project Defect Prediction; TNB, transfer naïve bayes; CCA, Canonical Correlation Analysis; WPDP, Within-Project Defect Prediction; HDP, Homogeneous Cross Project Defect Prediction.

H0: The performance of the two models is identical.

H1: The performance of the two models differs.

If P-value <0.05 then H0 is rejected.

When examining Recall for the proposed ensemble framework and Random Forest, the null hypothesis for WSR test is rejected, with a P-value of 0.00172. When AUC was taken into account for the proposed ensemble framework and Random Forest, similar results were obtained. The Ho was again discarded, with a P-value of 0.0004. The denied Ho claims that the two models work differently, and thus the two modes are distinct.

Conclusions

Cross-project defect prediction with heterogeneous metric sets has received little attention. A novel HCPDP ensemble architecture based on metric matching has been proposed in this work. An optimized approach to resolve the CIB issue in HCPDP has been implemented and validated. It's a decision-making tool for predicting software project defects using minimal historical data and a variety of metrics. Six open-source object-oriented projects have been used to test the architecture. The databases were highly unbalanced. The proposed ensemble framework is used for dual purposes in our scenario. First, it corrects the issue of CIB in the source dataset by balancing the number of defective and non-defective instances. The framework is then used to predict defect-prone classes in the second step. Based on our statistical review, the proposed system appears to be feasible and yields promising results. The framework's training accuracy was assessed using k-fold cross validation. We used the WSR test with a significance level of 5% (i.e., P-value=0.05) to demonstrate the validity of the proposed framework. We concluded that the proposed model is different from the baseline models.

Our future work will be focused on generalization of the proposed approach using more defect data with heterogeneous metrics and implementation of further complex learning approaches for defect predictions with a better performance.

Data availability

Underlying data

The datasets from the six open-source projects used in this study are freely available from OpenML:

PC2: https://www.openml.org/search?type=data&status=active&id=1069

PC3: https://www.openml.org/search?type=data&status=active&id=1050

PC4: https://www.openml.org/search?type=data&status=active&id=1049

AR3: https://www.openml.org/search?type=data&status=active&id=1060

AR4: https://www.openml.org/search?type=data&status=active&id=1061

AR6: https://www.openml.org/search?type=data&status=active&id=1064

Figshare: software defect prediction dataset. https://doi.org/10.6084/m9.figshare.20209142.v1.[34]

This project contains the following underlying data, which can be directly read using the python source code provided in *Software availability*:

- ar3.arff (class level defect data from the ar3 project).
- ar4.arff (class level defect data from the ar4 project).
- ar6.arff (class level defect data from the ar6 project).
- pc2.arff (class level defect data from the pc2 project).
- pc3.arff (class level defect data from the pc3 project).
- pc4.arff (class level defect data from the pc4 project).

Data are available under the terms of the Creative Commons Zero "No rights reserved" data waiver (CC0 1.0 Public domain dedication).

Software availability

Source code available from: https://github.com/lipika-amity/Heterogeneous-CPDP

Archived source code at time of publication: https://doi.org/10.5281/zenodo.6961342.[35]

License: Apache-2.0

References

1. Han D, Hoh IP, Kim S, *et al.*: *Micro interaction metrics for defect prediction, in Proceedings of the 16th ACM SIGSOFT international Symposium on Foundations of software engineering.* New York, USA: ACM; 2011.
2. D'Ambros M, Lanza M, Robbes R: **Evaluating defect prediction approaches: a benchmark and an extensive comparison.** *Empir. Softw. Eng.* 2012; **17**(4-5): 531–577.
Publisher Full Text
3. Goel L, Sharma M, Khatri SK, *et al.*: **An empirical analysis of the statistical learning models for different categories of Cross Project Defect Prediction.** *Int. J. Comput. Aided Eng. Technol.* 2021; **14**(2): 233.
Publisher Full Text
4. Canfora G, De Lucia A, Oliveto R, *et al.*: **Multiobjective cross-project defect prediction.** *IEEE Sixth International Conference on Verification*

and Validation in Software Testing. IEEE, Luxembourg, Luxembourg. ISSN 2159-4848. 2013.

5. Bener AB, Menzies T, Di Stefano J, *et al.*: **On the relative value of crosscompany and within-company data for defect prediction.** *Empir. Softw. Eng.* 2009; **14**: 540–578.
Publisher Full Text
6. Butcher A, Cok DR, Marcus A, *et al.*: **Local vs. global models for effort estimation and defect prediction.** *In 26th IEEE/ACM International Conference on Automated Software Engineering ASE 2011.* IEEE, Lawrence, KS, USA. pp. 343–3512011.
7. Camargo Cruz AE, Ochimizu K: **Towards logistic regression models for predicting fault- prone code across software projects.** *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM), Lake Buena Vista, Florida, USA.* 2009; pp. 460–463.
8. Briand LC, Melo WL, Wurst J: **Assessing the applicability of fault-proneness models across object-oriented software projects.** *IEEE Trans. Softw. Eng.* 2002; **28**: 706–720.
Publisher Full Text
9. Devanbu P, Posnett D, Rahman F: **Recalling the imprecision of cross- project defect prediction.** *In Proceedings of the ACM-Sigsoft 20th International Symposium on the Foundations of Software Engineering (FSE- 20).* ACM, Research Triangle Park, NC, USA. 2012; pp. 61–65.
10. Canfora G, De Lucia A, Oliveto R, *et al.*: **Multiobjective cross-project defect prediction.** *IEEE Sixth International Conference on Verification and Validation in Software Testing.* IEEE, Luxembourg, Luxembourg. ISSN 2159-4848. 2013.
11. Jing X, Wu F, Dong X, *et al.*: **Heterogeneous cross company defect prediction by unifiedmetric representation andCCA-based transfer learning.** *Proceedings of the 10th JointMeeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of SoftwareEngineering, ESEC/FSE 2015, ita.* September 2014; pp. 496–507.
12. He P, Li B, Ma Y: **Towards cross-project defect prediction with imbalanced feature sets.** *CoRR.* 2014; **abs/1411.4228**.
13. Jing X, Wu F, Dong X, *et al.*: **Heterogeneous cross company defect prediction by unifiedmetric representation and CCA-based transfer learning.** *Proceedings of the 10th JointMeeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of SoftwareEngineering, ESEC/FSE 2015, ita.* September 2015; pp. 496–507
14. Ryu D, Jang J-I, Baik J: **A transfer cost-sensitive boostingapproach for cross-project defect prediction.** *Softw. Qual. J* 2015; **25**(1): 235–272.
Publisher Full Text
15. Xinglong Yin X, Liu L: **Huaxiao Liu, Qi Wu. Heterogeneous cross-project defect prediction with multiple source projects based on transfer learning [J].** *Math. Biosci. Eng.* 2020; **17**(2): 1020–1040.
Publisher Full Text
16. Fu W, Kim S, Menzies T, *et al.*: **Heterogeneous defect prediction.** *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE, ACM, New York, NY, USA.* 2015; pp. 508–519.
17. Fu W, Kim S, Menzies T, *et al.*: **Heterogeneous defect prediction.** *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE, ACM, New York, NY, USA.* 2016; pp. 508–519.
18. Choosing software metrics for defect prediction: **An investigation: comparison and improvements.** *IEEE Access.* 2017; **5**: 25646–25656.
Publisher Full Text
19. Ni C, Liu W, Gu Q, *et al.*: **FeSCH: A FeatureSelection Method using Clusters of Hybrid-data for Cross-Project Defect Prediction.** *Proceedings of the 41st IEEE Annual Computer Software and Applications Conference, COMPSAC 2017, ita.* July 2017; pp. 51–56.
20. Xu Z, Yuan P, Zhang T, *et al.*: **HDA: Cross Project Defect Prediction via Heterogeneous Domain Adaptation With Dictionary Learning.** *IEEE Access.* 2018; **6**: 57597–57613.
Publisher Full Text
21. Gong L, Jiang S, Yu Q, *et al.*: **Unsupervised Deep Domain Adaptation for Heterogeneous Defect Prediction.** *IEICE Trans. Info. And Syst.* 2019; **E102.D**(3): 537–549.
Publisher Full Text
22. Sun Y, Jing X-Y, Fei W, *et al.*: **Semi-supervised Heterogeneous Defect Prediction with Open-source Projects on GitHub.** *Int. J. Softw. Eng. Knowl. Eng.* 2021; **31**(06): 889–916.
Publisher Full Text
23. Kim E, Baik J, Ryu D: **A Selection Technique of Source Project in Heterogeneous Defect Prediction based on Correlation Coefficients.** *J. KIISE.* 2021; **48**(8): 920–927.
Publisher Full Text
24. Chen L, Wang C, Song S: **Software defect prediction based on nested-stacking and heterogeneous feature selection.** *Complex IntellSyst.* 2022; **8**: 3333–3348.
Publisher Full Text
25. Goel L, Sharma M, Khatri SK, *et al.*: **A Framework for Homogeneous Cross Project Defect Prediction.** *Int. J. Softw. Innov.* 2020; **9**(1): 52–68.
Publisher Full Text
26. Goel L, Sharma M, Khatri SK, *et al.*: **Cross-project defect prediction using data sampling for class imbalance learning: an empirical study.** *Int. J. Parallel Emergent Distrib. Syst.* 2021; **36**(2): 130–143.
Publisher Full Text
27. Malhotra R, Kamal S: **An Empirical Study to Investigate Oversampling Methods for Improving Software Defect Prediction Using Imbalanced Data.** *Neurocomputing.* 2019; **343**: 120–140.
Publisher Full Text
28. Nandal N, Tanwar R, Pruthi J: **Machine learning based aspect level sentiment analysis for Amazon products.** *Spat. Inf. Res.* 2020; **28**: 601–607.
Publisher Full Text
29. Liaw A, Wiener M: **Classification and Regression by RandomForest.** *Forest.* 2001; **23**.
30. Chen T, Guestrin C: **XGBoost: A Scalable Tree Boosting System.** 2016; 785–794.
Publisher Full Text
31. Gao K, Khoshgoftaar TM, Wang H, *et al.*: **Choosing software metrics for defect prediction: An investigation on feature selection techniques.** *Softw Pract. Exper.* Apr. 2011; **41**(5): 579–606.
Publisher Full Text
32. Durango A, Refugio C: **An empirical study on Wilcoxon Signed Ranked Test.** 2018.
Publisher Full Text
33. Ampomah EK, Qin Z, Nyame G: **Evaluation of Tree-Based Ensemble Machine Learning Models in Predicting Stock Price Direction of Movement.** *Information.* 2020; **11**(6): 332.
Publisher Full Text
34. Goel L: **software defect prediction dataset. figshare [Dataset].** 2022.
Publisher Full Text
35. Lipika-amity: **lipika-amity/Heterogeneous-CPDP: (v1.0). Zenodo. [Software].** 2022.
Publisher Full Text