International Journal of e-Collaboration
Volume 19 • Issue 1

Addressing Noise and Class Imbalance Problems in Heterogeneous Cross-Project Defect Prediction:
An Empirical Study

Rohit Vashisht, Jamia Millia Islamia, New Delhi, India & KIET Group of Institutions, Delhi-NCR, Ghaziabad, India*
Syed Afzal Murtaza Rizvi, Jamia Millia Islamia, India

ABSTRACT

When a software project either lacks adequate historical data to build a defect prediction (DP) model or is in the initial phases of development, the DP model based on related source project's defect data might be used. This kind of SDP is categorized as heterogeneous cross-project defect prediction (HCPDP). According to a comprehensive literature review, no research has been done in the field of CPDP to deal with noise and class imbalance problem (CIP) at the same time. In this paper, the impact of noise and imbalanced data on the efficiency of the HCPDP and with-in project defect prediction (WPDP) model is examined empirically and conceptually using four different classification algorithms. In addition, CIP is handled using a novel technique known as chunk balancing algorithm (CBA). Ten prediction combinations from three open-source projects are used in the experimental investigation. The findings show that noise in an imbalanced dataset has a significant impact on defect prediction accuracy.

KEYWORDS

Classification, Cross-Project, Defect, Heterogeneous, Imbalance, Noise, Regression, With-In

INTRODUCTION

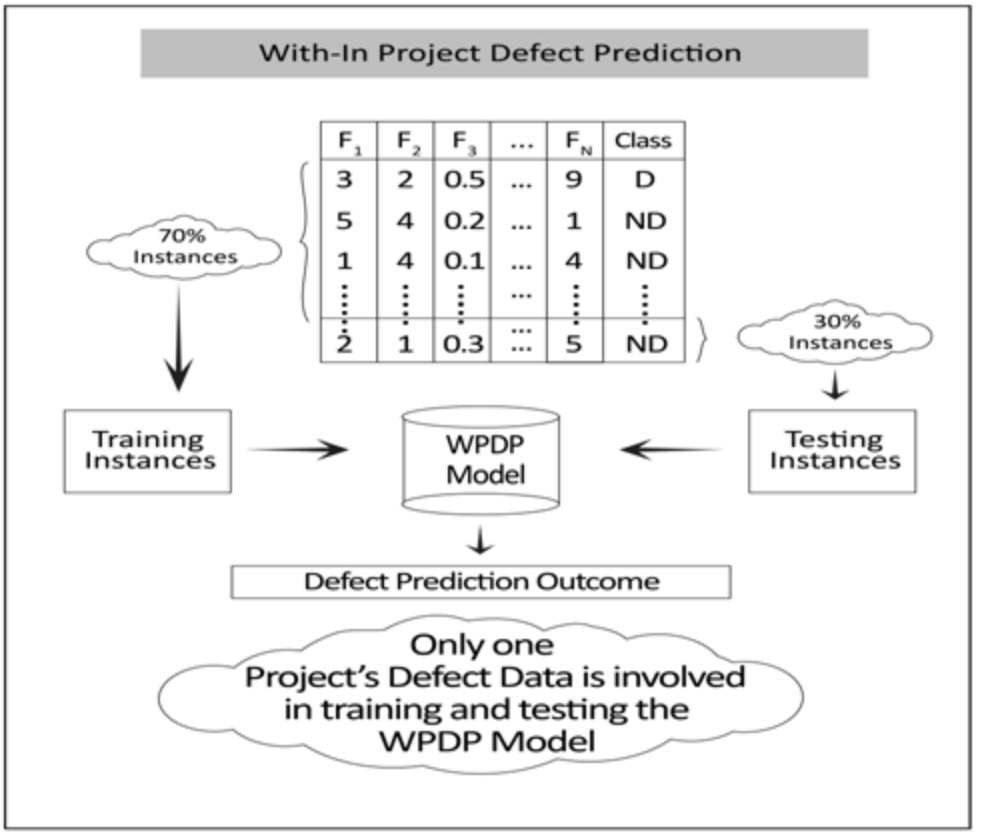
Software has become an essential part of everyone's daily life in today's digital era. Even a minor flaw or malfunction in this software might result in financial or even life-threatening losses. Inconsistencies, ambiguities or misinterpretation of the specifications, carelessness or negligence in writing code, insufficient testing, unsuitable or unanticipated use of the software, or other unforeseen issues can all cause software errors. Software testing should be done at the proper time in the early stages of Software Development Life Cycle (SDLC) in order to reduce overall software development cost. The SDLC software testing phase, on the other hand, accounts for 60% of the total cost of software development. As a result, it's vital to do testing on the appropriate modules at the appropriate time.

Software Defect Prediction (SDP) can be broadly split into two classes, according to the state of the art: Within Project Defect Prediction (WPDP) and Cross Project Defect Prediction (CPDP). The available defect dataset is split into two parts in WPDP in order to build the DP model in such a way that one half of the dataset (referred to as labeled observations) is used to train the DP model and the other portion is used to validate the DP model, as illustrated in Figure 1.Finding labels that

DOI: 10.4018/IJeC.315777 *Corresponding Author

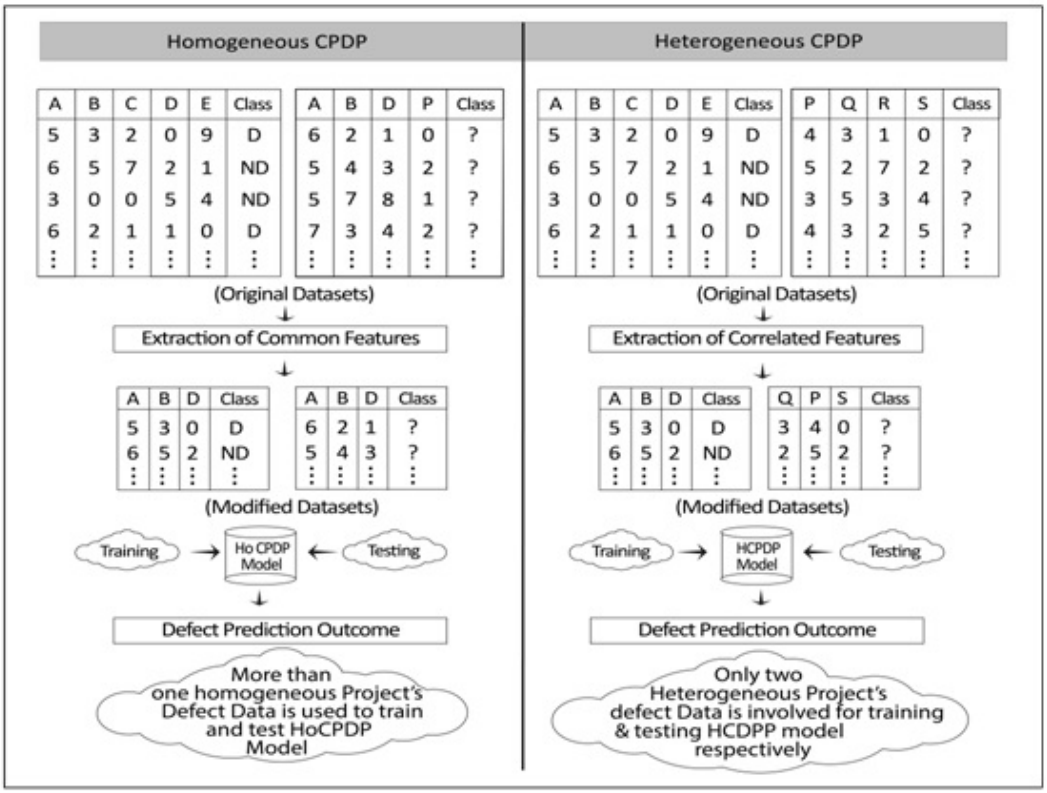
are either faulty or non-faulty for unidentifiable instances in the target dataset is how the DP model is tested (Ambros et al., 2012).

Figure 1. With-In project defect prediction



CPDP is a type of SDP in which software projects that lack the required local defect data can develop an accurate and effective DP model using data from other projects. CPDP can also be divided into two subcategories: Homogeneous CPDP (HoCPDP) and Heterogeneous CPDP (HCPDP). HoCPDP collects common software measures/features from both the source (whose defect data is used to train the SDP model) and the target (for which the SDP model is created) applications (He et al., 2014). When using HCPDP, however, there are no uniform metrics between the prediction pair datasets. Uniform features between two applications can be determined by evaluating the coefficient of correlation between all possible software feature combinations. In the case of HCPDP, combinations of feature pairs with a similar distribution in their values are employed as common features between source and target datasets in order to forecast project-wide problems. As shown in Figure 2, correlated feature pairs for the HCPDP category include (A, Q), (B, P), and (D, S). Figure 2 provides more details on both CPDP groups.

Figure 2. Categories of cross project defect prediction



The change-log documents of software configuration management may be used in SDP models, as the change-log does indeed report the modules that change when defects are recognized and corrected. In defect datasets, there are two key limitations that might contribute to poor SDP model prediction performance. Firstly, due to a number of factors, the linkages between logs and bug reports may be unreliable, resulting in mislabeled data (Kurt et al., 1999). As a result, it's very likely that an SDP model is operating with noisy data and producing incorrect findings. Secondly, in a binary classification problem, where the cardinality of instances for one kind of class is very low compared to the other, the model developed from such data produces skewed and biased results in favor of the majority class. In training datasets, there are a variety of strategies for dealing with CIP. There are two strategies to equalize the number of cases in the majority and minority classes if data resampling techniques (Marques et al., 2013) are used to combat CIP. However, resampling techniques have drawbacks that will be explored in the later section. To overcome the drawbacks of sampling strategies, a novel technique called Chunk Balancing Algorithm (CBA) is utilized in this paper to induce Class Imbalance Learning (CIL) (Vashisht & Rizvi, 2021).

According to a thorough review of the literature, no research has been done to address both noise and CIP in the context of CPDP. The main objective of the proposed research study is to evaluate the efficacy of the HCPDP model with varied levels of noise and imbalanced datasets. The primary contributions of the paper are as follows: -

RQ1. To compare the prediction performance of traditional method of SDP i.e., WPDP at various levels of noise with or without handling of CIP.

RQ2. To compare the prediction performance of HCPDP at various levels of noise with or without handling of CIP.
RQ3. To determine the maximum level of noise that each prediction pair can tolerate under both SDP categories (HCPDP & WPDP).
RQ4. Which classification method outperforms the rest of the algorithms in use?

The proposed research study is organized as follows: - section B provides a detailed literature survey on CPDP, section C describes the four-phase HCPDP model, three phase WPDP model and the CBA approach used in the proposed research study; section D summarizes the datasets used for analysis of the proposed work and the performance parameters used to evaluate the experimental results. The experimental set up and results are discussed in section E & F respectively and the final conclusions are reported in section G.

STATE OF THE ART

Melo et al. (2002) reported the first recognized study in CPDP. In two Java-based frameworks, Xpose and Jwriter, they established the MARS (Multivariate Adaptive Regression Spline) paradigm for defect detection and data design. They forecast the classes in Jwriter using their tendency for fault. They achieved this by training a model on the Xpose dataset. They evaluated MARS' efficiency to that of Linear Regression (LR) and determined that MARS beats LR and is considerably cheaper.

Menzies et al. (2009) used data from ten projects from two sources. They remove noisy, repetitive, and irrelevant data from the data and train the model with this unblended data for successful defect prediction. On data from ten projects, the tests were conducted using the Nearest Neighbor (NN) approach. The results revealed that the tests were effective at predicting project faults. Meanwhile, the CPDP job was unable to outperform the project defect prediction task in these studies.

Camargo et al. (2009) employed log transformation for the first time in the same year to detect related instances in training and analyzing project data in order to remove project-based data instances. In the same year, Menzies et al. (2009) proposed classifying defect prediction on Internet Explorer and Mozilla Firefox as training and testing initiatives. They employed the coding model and process parameters to complete the classification job. They trained the suggested DP model using Mozilla Firefox defect data, which was subsequently utilized to forecast problems in Internet Explorer. When the suggested model was utilized as a training project and Mozilla Firefox was employed as a testing project, these studies demonstrated that the proposed model outperformed.

Menzies et al. (2011) claimed that relevance varies depending on how it is interpreted. They claimed that the meaning of data varies based on how it is interpreted, and that data relevance can be contradictory depending on how it is interpreted. Data that appears significant when seen worldwide can be useless when viewed locally. They backed up their ideas with studies, concluding that local behaviour outperformed global behavior and that condition-based laws should take precedence over other considerations.

Bellenburg et al. (2012) expanded to Menzies et al. (2011) arguments by proving that while local models were better for a specific dataset, global models were better for generality. Rahman et al. (2012) did research in the same year to show that performance metrics like F-score, accuracy, and recall are insufficient for quality assurance when defect prediction is made using multiple models. According to them, AUC produces equivalent results in WPDP models. To overcome the drawbacks of the single-objective model (Rahman et al., 2012), Canfora et al. (2013) proposed a multi-objective strategy. To practise the Logistic Regression (LR) model, they employed a non-dominated sorted Generic Algorithm (NSGA-II).

Gao et al. (2011) used 1398 projects from Google code and source forge to create a Universal Defect Prediction (UDP) model. This model compares the metrics in the datasets of the training and testing projects, and if at least 26 of them match, only predictions on the target project may be produced.

He et al. (2014) developed a novel metric based on instance characteristic vectors to overcome this constraint. When comparing CPDP to feature disparity, they discovered negative consequences. Three separate datasets were used to test 11 different projects.

To model faults, Dong et al. (2015) suggested utilizing Canonical Correlation Analysis (CCA). They were the first to bring the concept of Heterogeneous Defect Prediction (HDP) to the attention of the general audience. They eliminate the metrics disparity issue between the training and testing project datasets by adding dummy metrics with null values. They put 14 projects to the test using four different datasets.

Ni et al. (2017) developed FESCH, a unique strategy that outperformed TCA+ and WPDP in most circumstances while also providing state-of-the-art results for the baseline methodologies used. FeSCH's success was also self-sustaining and independent of the classifiers utilized, according to the data.

Li et al. (2017) examined the four filtration procedures for defect data in the same year. According to them, the fault data filtration method adopted has a big impact on the model's capacity to anticipate problems. Four different filtering approaches were compared: Local Cluster based Filter (LCBF), Data Characteristic based Filter (DCBF), Target project data Guided Filter (TGF), Source project data Guided Filter (SGF), and Target project data Guided Filter (TGF) (LCBF). They also added a new filter, the Hierarchical Selection Dependent Filter (HSDF), to address the scalability issues with the other four filters when working with large datasets. Existing filtering strategies were outperformed by the suggested filtering strategy.

After gathering similar data from prior versions of the same system, Lee and Felix (2020) focused on method-level (ML) defect estimation using regression models in a recent software release. The authors employed three performance assessment factors to implement defects that demonstrate a substantial association with ML defects, such as defect density, defect velocity, and time. The proposed study and evaluation of pre-to-post system data pre-processing classifiers and entropy values in average output datasets was also made easier to the proposed effort. With a 93 percent connection, defect velocity demonstrated the strongest correlation with the count of ML faults of all three components.

Deep learning methods were suggested by Majd et al. (2020) for forecasting Statement Class Defects (SCD) in the same year. By specifying regions or modules that are more prone to errors, the authors of this research hoped to alleviate the burden on software engineers. The authors ran trials on 1,19,989 C/C++ programs using Code4Bench's Broad Short-Term Memory (BSTM) deep learning model. The SCD model was put to the test for anticipating faults in unknown data (i.e., new statements), and the authors discovered that it worked well, with high memory, precision, and accuracy.

Grassmann Manifold Optimal Transfer Defect Prediction (GMOTDP) is a novel work in the subject of HCPDP that was presented in the year 2020. Jiang et al. (2020) proposed a three-phase HDP model that included a Mahalanobis distance-based Class Imbalance Learning (CIL) framework for dealing with Class Imbalance Problem (CIP) in the source dataset, as well as a CART-based ensemble learning methodology for finding the best subset of the source dataset for metric matching. The authors examined nine projects from three public domain software defect repositories to four recognized advanced methodologies to see if the proposed approach in this paper was feasible. In terms of AUC, the results of the experiments suggest that the proposed strategy is more reliable.

Diwaker et al. (2019) introduced a novel model for estimating the reliability of component-based software (CBS) utilizing series - parallel reliability theories, and then tested the developed component-based software reliability concept using two soft computing techniques: Fuzzy Logic and PSO. In comparison to reliability prediction using fuzzy logic and PSO, the experimental findings show that the suggested reliability model has a lower error rate in forecasting CBSE reliability.

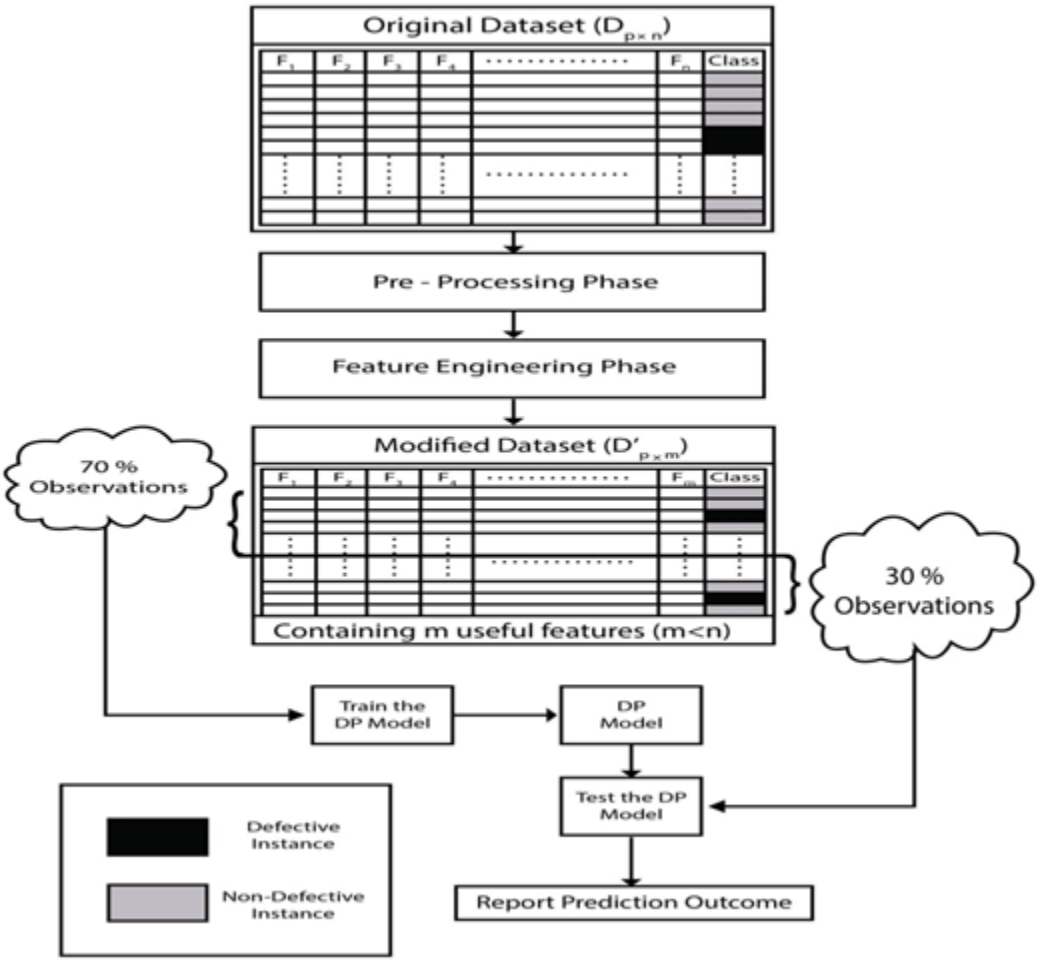
Panwar et al. (2019) provides a new optimal mathematical model for predicting stakeholder satisfaction levels (Q). The relationship implications of various quality attributes are used to validate the real data in optimal models. It employs constraint equations. Q's maximum and minimum values are determined by the best model. Constraints are features of software quality.

Preprocessing, Deep Learning Model, and Duplicate Bug Report Detection and Classification are the three elements of the proposed system by Kukkar et al. (2020). In addition, the suggested technique uses a deep learning model based on Convolutional Neural Networks to extract significant features. These relevant features are used to identify bug report features that are similar. As a result of these identical traits, the bug reports are similar to computers.

PROPOSED WORK

In this study, two models for each of the SDP categories are proposed. Firstly, WPDP, the conventional DP technique, employs only one dataset, which is subsequently partitioned into two components, one for training and one for testing, according to the partition strategy. Figure 3 depicts the three-phase WPDP model in detail.

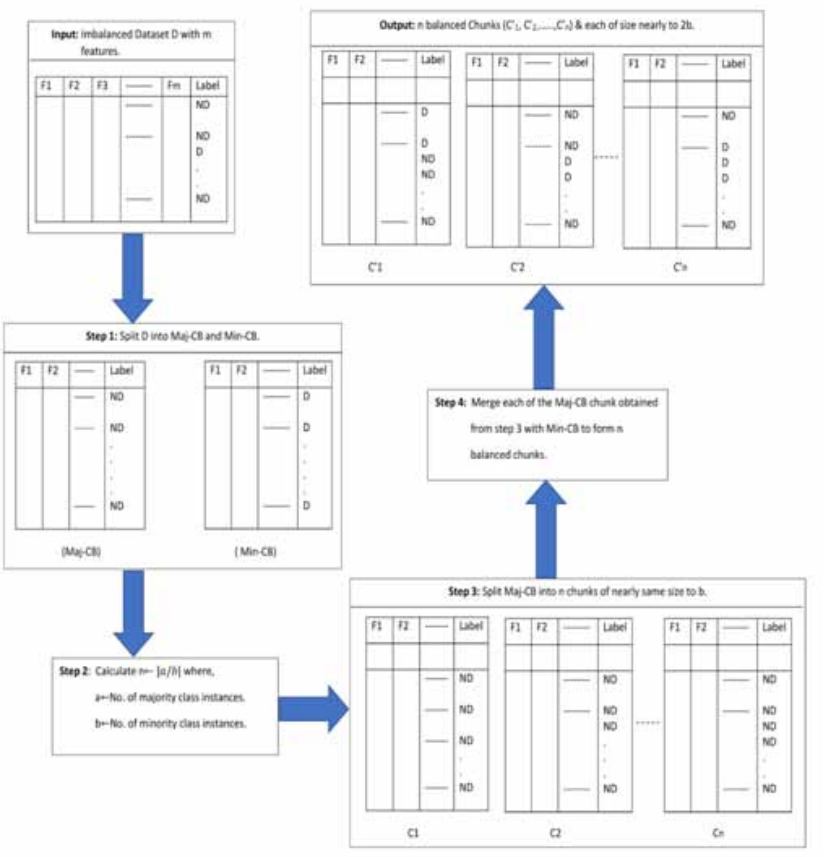
Figure 3. Three phase WPDP model



Preprocessing datasets is done in the initial phase to ensure that they are suitable to be used in any machine learning model. It encompasses dealing with missing values, labeling the categorical variable, and dealing with CIP if any of the training datasets contain it. The most significant challenge to be addressed in this phase is CIP. Resampling is the data-based approach which tries to balance the count of instances either by eliminating the majority class instances called Random Under Sampling Technique (RUST) or by increasing the minority class instances called Random Over Sampling Technique (ROST) (Vashisht & Rizvi, 2020).

However, both techniques have limitations, according to the literature survey done for the proposed analysis (Vashisht & Rizvi, 2020). ROST method generates duplicate minority class observations, which can over-generalize the minority class without considering the distribution of instances in the majority class. RUST, on the other hand, can ignore certain useful or relevant observations in the majority class without considering their importance in predicting the future outcome. As a result, to handle the imbalance count of observations in datasets, a novel CIL approach called CBA is used (Vashisht & Rizvi, 2021). In Figure 4, the CBA concept is well-explained.

Figure 4. Chunk balancing algorithm



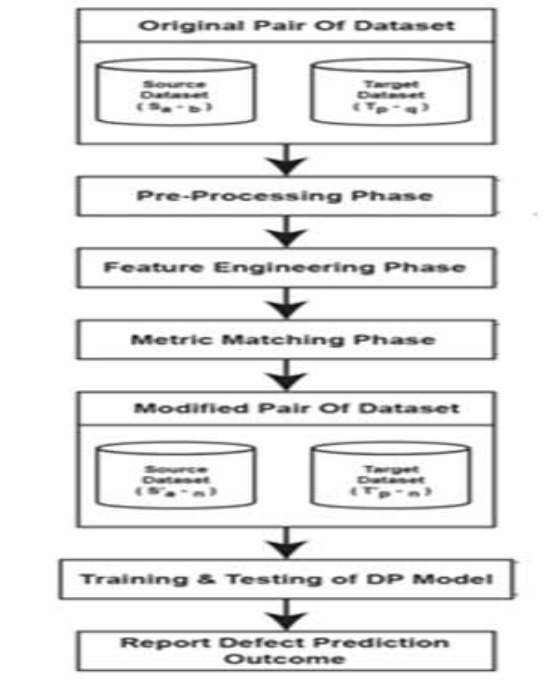
The approach starts with an imbalanced input dataset D and has m software features. In the first step, D is divided into two classes: Majority Class Bunch (Maj-CB) and Minority Class Bunch (Min-CB), each with a and b number of observations respectively. Non-defective and defective classes are

represented by Maj_CB and Min_CB, respectively. CBA then calculates n and divides Maj-CB into n chunks of nearly identical size as b. In the last step, each majority class chunk created in step 3 is merged with Min_CB to produce n balanced chunks (C'_1, C'_2,....., C'_n) with a size near to 2b. After preprocessing step, second phase of WPDP modeling i.e., Feature Engineering (FE), redundant and worthless features are eliminated from the original datasets, and the top n features are chosen using an appropriate feature selection technique (Mwadulo et al., 2015). WPDP divides the given dataset into training and validation datasets in a 7: 3 ratio, as per the state of the art. In the final phase, the performance efficiency of WPDP model is evaluated on the basis of machine learning classification algorithms.

While HoCPDP permits many similar project datasets to be used for training and testing the DP model, HCPDP starts the DP process with a pair of datasets known as the source and the target datasets as shown in Figure 5. In both datasets, each row and column represents an instance and a software metric, respectively. The pre-processing phase and the FE phase of HCPDP model are implemented in the same way as they are executed in WPDP model.

The third phase of heterogeneous prediction modeling, metric matching, is the most crucial and challenging stage since the model's training accuracy is mostly relied on the matched metric collection. Modern methods such as the least square method, dispersion diagram method, and Spearman's Rank Correlation Method (SRCM) can be used to display the relationship between metrics. The model picks those feature pairs whose Coefficient of Correlation Value (CCV) is greater than the chosen cutoff level after evaluating CCV between different possible feature pairs of two projects. The set of features generated after applying a cutoff filter is known as strongly correlated features. If this highly correlated metric collection for a pair of datasets (S, T) is null, the source dataset S cannot be used to model defects in a heterogeneous target dataset T. After discovering this highly correlated metric collection, the model is trained using an appropriate machine learning method, and performance results are given in the model's final step. The output findings are summarized using various assessment measures.

Figure 5. Four phase HCPDP model



DATASETS USED & PERFORMANCE MEASURES

Datasets Description

This section goes over the datasets that were used in the analysis. 13 benchmarked datasets from the AEEEM, ReLink, and SOFTLAB repositories are included in the study. Object-oriented metrics (OOs), past defect metrics, application metrics, and other metrics are included in the AEEEM collection. The Understand tool produced 26 coding consequences findings, which were saved in the ReLink repository. SOFTLAB also has patented data sets that include cyclomatic measurements from Halstead and McCabe.

The Class Imbalance Ratio (CIR) is the ratio of defective to non-defective occurrences, or the other way around. The greater the Imbalance Problem (IP) in a particular training sample, the lower its CIR value. In the datasets ar1 and Apache, CIR values vary from 7.43 (highest IP) to 102.08 (lowest IP), respectively. In three repositories, however, there are 61, 26 and 29 software metrics, respectively. The following is the source of all datasets:

https://github.com/bharlow058/AEEEM-and-other-SDP-datasets/tree/master/dataset

The statistics of CIR in all three software repositories are depicted in Figures 6, 7, and 8. CIR is highest in datasets EQ (66.15), Apache (102.08), and ar5 (28.57), indicating that they have the least or no CIP among the datasets in their respective groups. The datasets LC (10.2), ZXing (41.99), and ar1 (7.43), on the other hand, have the lowest CIR, implying that there is a higher need to address CIP in these datasets.

Additional information about these datasets can be found in Table 1. In three project categories, the proportion of defective cases ranges from 7.43 percent to 50.51 percent, according to Table 1.

Table 1. Datasets description

Project Group	Datasets	Count of Observations			Count of Software Metrics
		Total	Defective	Non-Defective	
AEEEM	EQ	324	129 (39.81%)	195 (60.19%)	61
	JDT	997	206 (20.66%)	791 (79.34%)	
	LC	691	64 (9.26%)	627 (90.74%)	
	ML	1862	245 (13.15%)	1617 (86.85%)	
	PDE	1492	209 (14.01%)	1283 (85.99%)	
ReLink	Apache	194	98 (50.51%)	96 (49.49%)	26
	Safe	56	22 (39.28%)	34 (60.72%)	
	Zxing	399	118 (29.57%)	281 (70.43%)	

Table 1 continued on next page

Table 1 continued

Project Group	Datasets	Count of Observations			Count of Software Metrics
		Total	Defective	Non-Defective	
SOFTLAB	ar1	121	9 (7.43%)	112 (92.57%)	29
	ar3	63	8 (12.69%)	55 (87.31%)	
	ar4	107	20 (18.69%)	87 (81.31%)	
	ar5	36	8 (22.22%)	28 (77.78%)	
	ar6	101	15 (14.85%)	86 (85.15%)	

Figure 6. CIR in AEEEM project group

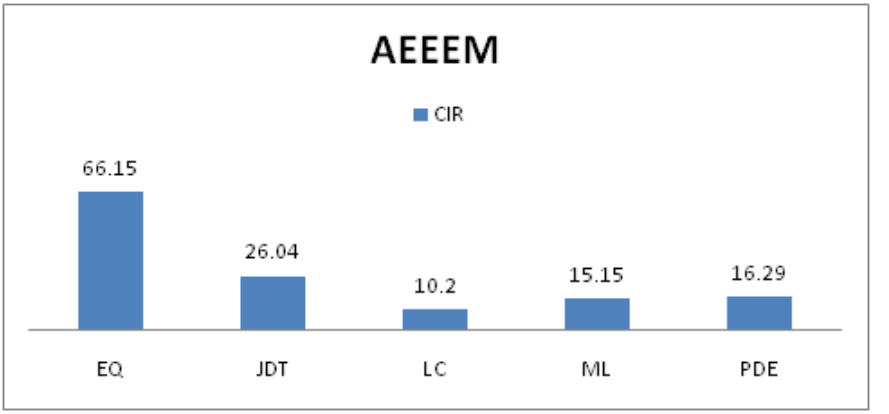


Figure 7. CIR in ReLink project group

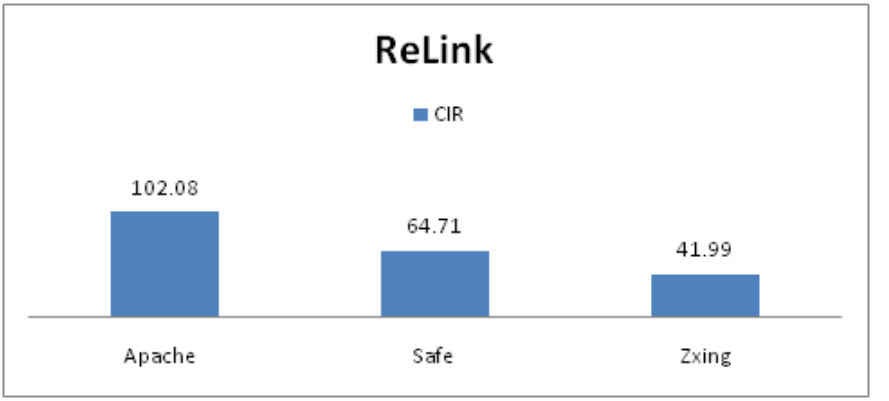
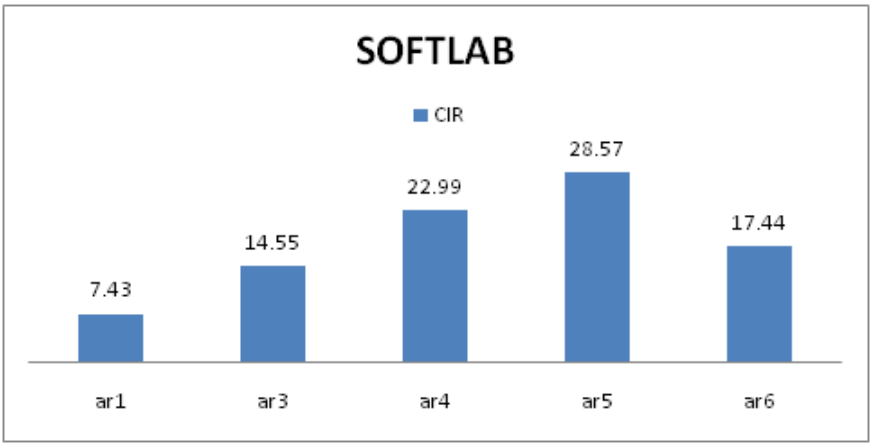


Figure 8. CIR in SOFTLAB project group



Performance Parameters

The numerous measures used to assess the efficacy of different machine learning classifiers are listed in this section. The additional factors in the confusion matrix are considered during the evaluation process. The confusion matrix used to estimate erroneous classifications is shown in Table 2.

Table 2. Confusion matrix

		Predicted Result	
		Defective	**Non- Defective**
Actual Result	Defective	True Positive (TP)	False Negative (FN)
	Non- Defective	False Positive (FP)	True Negative (TN)

- Recall: - It's also known as sensitivity or true positive rate. It genuinely tells you what percentage of actual positives was correctly identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

- False Positive Rate (FPR): - It's also known as the fall rate. It actually tells you what percentage of negative cases in the data was misclassified as positive.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2}$$

- F1-Score: - Because it is a measurement of a test's accuracy in a statistical study of binary classification, it is also known as the F-measure. It considers both the test's accuracy and precision when calculating the score. According to the eq (3), the harmonic mean of precision (p) and recall (r) is utilized to determine it.

$$\text{F-Score} = \frac{2^{*}\text{p}^{*}\text{r}}{\text{p}+\text{r}} \tag{3}$$

- Area Under Curve (AUC): - A plot of True Positive Rate (TPR) and False Positive Rate (FPR) is used to measure a classification algorithm's overall efficacy. If the AUC parameter is adjusted to a higher value, the classification model will be more accurate. The maximum AUC value for a classification algorithm is 1.
- Accuracy: - Accuracy is defined as the ratio of true outcomes (TP and TN) to the total number of occurrences evaluated. Its value ranges from 0 to 1, with 0 being the least accurate outcome and 1 denoting the most accurate outcome.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{4}$$

EXPERIMENTATION SETUP

The proposed research study's principal objectives are grouped into four categories. The study's first two objectives assess the performance of the respective HCPDP and WPDP frameworks with and without handling CIP in datasets with varying degrees of noise introduced manually. The study's third goal is to determine the maximum degree of noise that a given prediction pair may tolerate for a specific classification algorithm for both SDP categories. The research analysis' final goal is to identify the best classification algorithm that outperforms all other classifiers, as well as to compare the prediction performance of WPDP and HCPDP for the set of classifiers used. In order to answer the four research questions, the research analysis conducted two experiments.

Experiment 1

The goal of this experiment is to compare the output of a traditional DP, specifically WPDP, with or without handling CIP at various level of noise. To conduct both experiments, noise levels of 0%, 15%, 30%, 45%, 60%, and 75% have been used. The 0% noise indicates that all instances of the datasets are correctly labeled. On the other hand, 60 percent noise suggests that 60 percent of the total instances are erroneously labeled, implying that their labels have been manually changed to the opposite label. To begin, the dataset is preprocessed to remove any superfluous software features and the categorical data is encoded using the tag. The defective and non-defective examples are assigned 0 or 1 in this process. To cope with CIP in imbalanced training datasets, this experiment also employs the novel CBA discussed in section C. Chi-Square Test (CST) is used as a feature selection technique for identifying the most important features that are significant to predict the expected outcome.

The available number of instances is divided into training and validation instances in a ratio of 7:3 after the most discriminating features are chosen. Figure 9 shows DP within a project, with I_Total, I_Train, and I_Test denoting the total, training, and testing observations in the dataset, respectively. The training and testing datasets involved in this experiment are shown in Table 3. Four different classifiers are used to evaluate the prediction accuracy of the WPDP model: Naive Bayes (NB), Support Vector Machine (SVM), Adaptive Boosting (AdaBoost), and Random Forest (RF).

Figure 9. With-In project defect prediction

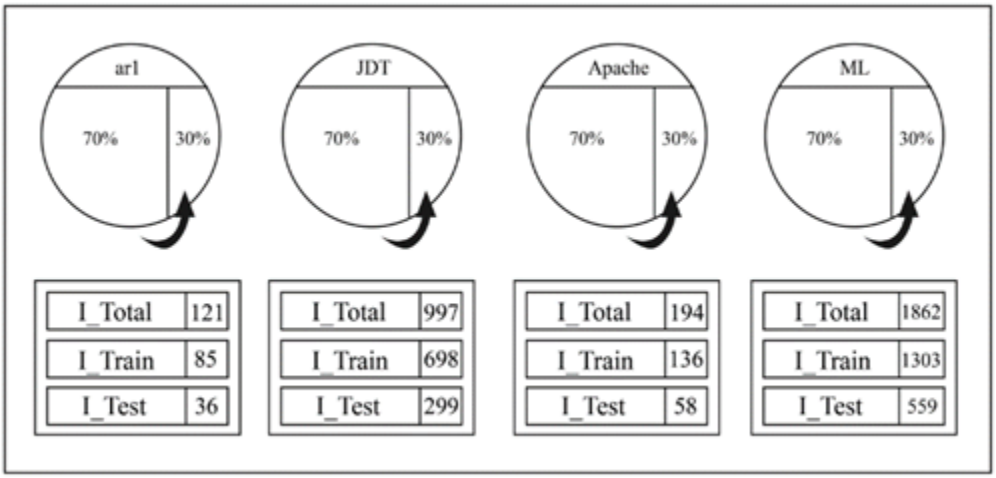


Table 3. WPDP's prediction combinations

Prediction Combination	Training Dataset (70%)	Testing Dataset (30%)
WPDP-C1	ar1	ar1
WPDP-C2	JDT	JDT
WPDP-C3	Apache	Apache
WPDP-C4	ML	ML

Experiment 2

The objective of this experiment is to explore how the preprocessing step affects the performance of the proposed four-phase HCPDP framework when CBA is used to induce CIL in imbalanced datasets. The six heterogeneous prediction combinations listed in Table 4 were obtained from three open-source projects: AEEEM, ReLink, and SOFTLAB, respectively. The number of maximal associated feature pairs found between them is used to generate prediction combinations. In the first phase of preprocessing datasets, redundant software features are removed, and categorical data is encoded with labels. Then, for a given dataset, CST is utilized as a feature ranking and feature selection approach to create a list of K- best features that are more relevant to the final outcome. Following the selection of useful features, the metric matching method assesses the relationship between each pair of source and target dataset feature pairs using SRCM (Mwadulo et al., 2015). The advantage of utilizing SRCM is that it considers the data's relevance, allows for further research, and does not presume normal distribution.

This experiment uses the same set of classification methods that were used in experiment 1 to train the HCPDP model, i.e., to carry out the final modeling step. Finally, performance parameters stated in section D are used to evaluate the model's efficiency in this experiment.

Table 4. HCPDP's prediction combinations

Prediction Combinations	Source Dataset	Target Dataset
HCPDP-C1	JDT	ar1
HCPDP-C2	ar5	JDT
HCPDP-C3	ar1	Apache
HCPDP-C4	Safe	ML
HCPDP-C5	EQ	ar3
HCPDP-C6	LC	ar3

RESULTS & DISCUSSION

Both experiments have been run on a Windows 10 operating system with an Intel Core i5-1130G7 processor and 32GB RAM, using TensorFlow 2.0 with GPU support. In this section, the findings of the experiments are discussed. The experimental results are shown in Tables 5 to 13 and Figures 10 to 17. FPR, recall, F-Score, and AUC are used as performance benchmarks in this study. The prediction's accuracy is measured using 10-fold cross-validation.

RQ1. To compare the prediction performance of traditional method of SDP i.e., WPDP at various levels of noise with or without handling of CIP.

To investigate the effectiveness of WPDP, four prediction combinations, WPDP_C1 to WPDP_C4, are considered, with 70% of the total cases being used to train the DP model and the remaining 30% being utilized to validate it. For example, there are 194, 136, and 58 as total, training and testing instances in WPDP_C3 (Apache dataset from ReLink source), as shown in Figure 9. Table 5 shows the FPR value for all prediction combinations with and without managing CIP at various levels of induced noise. Table 5 shows that WPDP_C3 with a CIR of 102.08 has the best performance, indicating that Apache has nearly equal numbers of instances from the majority and minority classes. For the RF and NB classification algorithms, the lowest and highest FPR values for WPDP_C3 are 0.13/0.23 (at 0% level of noise) and 0.58/0.65 (at 75% level of noise), respectively. Similarly, using the RF and AdaBoost classification approaches, the lowest and greatest FPR values for WPDP_C1 are 0.15/0.36 (at 75 percent level of noise) and 0.48/0.53 (at 15 percent level of noise), respectively. The reason for the higher disparity in FPR values for WPDP_C1 with and without managing CIP is that it has the highest CIR of 7.43 among all prediction combinations.

Table 6 reveals that for WPDP_C3, the maximum and minimum TPR values for RF and AdaBoost approaches are 0.97/0.94 (at 0% level of noise) and 0.51/0.49 (at 75% level of noise), respectively.

Table 5. FPR values at different level of noise (WPDP)

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
WPDP-C1	NB	0.36/0.39	0.34/0.37	0.34/0.39	0.35/0.36	0.37/0.40	0.51/0.68
	SVM	0.56/0.65	0.61/0.64	0.47/0.58	0.41/0.49	0.42/0.49	0.38/0.46
	AdaBoost	0.46/0.58	0.48/0.53	0.47/0.54	0.43/0.49	0.46/0.48	0.41/0.44
	RF	0.20/0.41	0.20/0.46	0.17/0.41	0.18/0.36	0.25/0.38	0.35/0.56
WPDP-C2	NB	0.31/0.47	0.23/0.29	0.33/0.29	0.22/0.34	0.28/0.27	0.30/0.39
	SVM	0.77/0.84	0.68/0.77	0.59/0.68	0.45/0.58	0.39/0.49	0.31/0.44
	AdaBoost	0.47/0.59	0.50/0.55	0.52/0.61	0.41/0.63	0.40/0.38	0.40/0.65
	RF	0.22/0.32	0.19/0.32	0.18/0.31	0.23/0.31	0.29/0.38	0.32/0.49
WPDP-C3	NB	0.32/0.39	0.35/0.35	0.45/0.47	0.49/0.55	0.52/0.59	0.58/0.65
	SVM	0.39/0.37	0.37/0.42	0.47/0.51	0.54/0.56	0.55/0.57	0.53/0.60
	AdaBoost	0.33/0.51	0.24/0.29	0.29/0.34	0.25/0.41	0.49/0.54	0.53/0.64
	RF	0.13/0.23	0.17/0.33	0.17/0.35	0.23/0.36	0.30/0.37	0.34/0.41
WPDP-C4	NB	0.27/0.28	0.27/0.30	0.31/0.37	0.29/0.34	0.35/0.39	0.30/0.38
	SVM	0.23/0.27	0.15/0.28	0.17/0.39	0.10/0.35	0.19/0.49	0.39/0.57
	AdaBoost	0.21/0.24	0.28/0.32	0.30/0.36	0.46/0.48	0.45/0.47	0.47/0.51
	RF	0.11/0.21	0.11/0.23	0.12/0.26	0.11/0.26	0.18/0.34	0.23/0.41

Table 6. TPR values at different level of noise (WPDP)

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
WPDP-C1	NB	0.57/0.53	0.51/0.48	0.56/0.50	0.55/0.59	0.61/0.65	0.40/0.33
	SVM	0.64/0.69	0.65/0.64	0.55/0.62	0.56/0.61	0.54/0.57	0.53/0.58
	AdaBoost	0.73/0.71	0.70/0.68	0.66/0.66	0.64/0.64	0.65/0.64	0.60/0.55
	RF	0.82/0.74	0.82/0.76	0.84/0.75	0.81/0.77	0.72/0.71	0.64/0.55
WPDP-C2	NB	0.72/0.71	0.73/0.70	0.71/0.68	0.72/0.70	0.68/0.69	0.65/0.62
	SVM	0.77/0.72	0.85/0.72	0.83/0.69	0.75/0.67	0.77/0.67	0.74/0.59
	AdaBoost	0.78/0.75	0.72/0.71	0.77/0.71	0.65/0.70	0.64/0.68	0.65/0.61
	RF	0.89/0.78	0.72/0.71	0.71/0.70	0.68/0.70	0.69/0.68	0.65/0.61
WPDP-C3	NB	0.85/0.92	0.81/0.80	0.70/0.63	0.69/0.58	0.61/0.56	0.63/0.52
	SVM	0.91/0.94	0.86/0.83	0.79/0.76	0.74/0.67	0.66/0.58	0.61/0.48
	AdaBoost	0.80/0.93	0.87/0.84	0.88/0.76	0.86/0.66	0.58/0.56	0.51/0.49
	RF	0.97/0.94	0.94/0.85	0.92/0.71	0.87/0.67	0.86/0.67	0.84/0.62

Table 6 continiued on next page

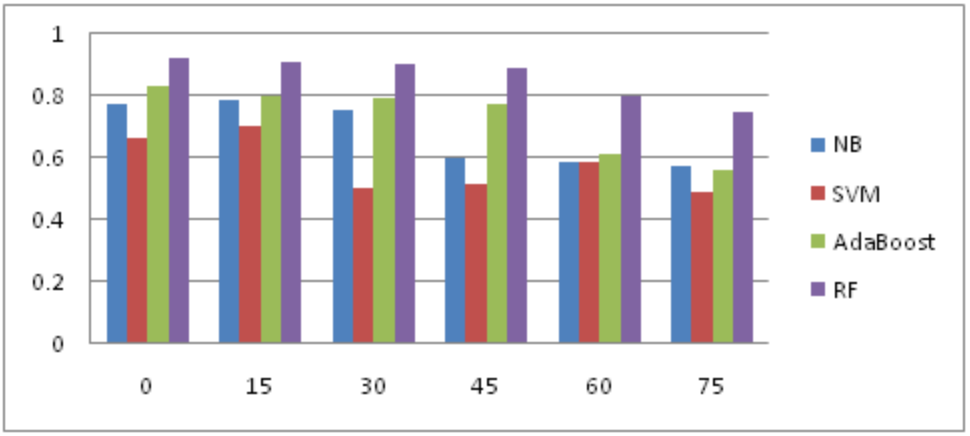
Table 6 continued

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
WPDP-C4	NB	0.67/0.63	0.61/0.58	0.66/0.60	0.65/0.69	0.66/0.62	0.66/0.63
	SVM	0.74/0.70	0.68/0.61	0.67/0.62	0.66/0.61	0.64/0.57	0.63/0.58
	AdaBoost	0.73/0.72	0.69/0.68	0.64/0.64	0.61/0.61	0.65/0.61	0.58/0.54
	RF	0.89/0.77	0.89/0.76	0.88/0.75	0.89/0.76	0.68/0.54	0.62/0.49

Figure 10 illustrates the average prediction accuracy for all WPDP prediction combinations. Practically, for all classification algorithms, there is a gradual decline in accuracy value as the level of induced noise increases. The accuracy value is collected using 10 fold cross validation to avoid any randomness in the results. The following is the increasing order of different classification algorithms based on calculated accuracy through repeated experimental results: -

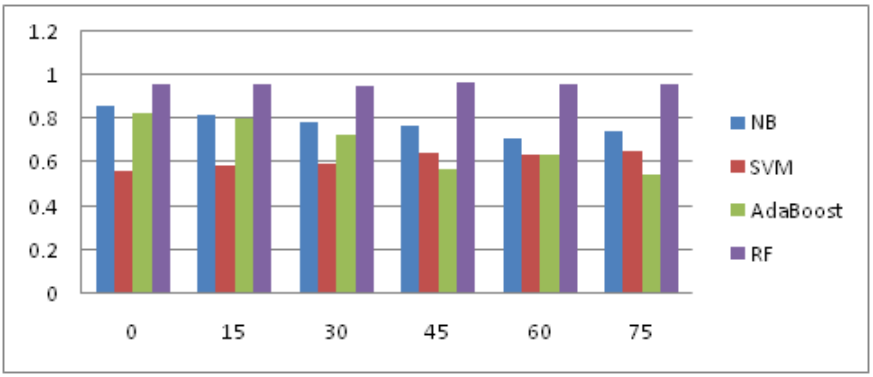
SVM < AdaBoost < NB < RF

Figure 10. WPDP's prediction accuracy



As demonstrated in Figure 11, a similar pattern of performance may be noticed in terms of AUC value. Even when the noise level is increasing, RF continues to provide consistent performance, as shown in the graph. Other classifiers exhibit a minor increase or decrease in AUC values when the induced noise level changes.

Figure 11. WPDP's AUC plot



In Figure 12 to Figure 15, the AUC plot for all prediction combinations has been seen. With the exception of RF in WPDP C2, it can be concluded that induced noise has a significant impact on AUC values for all classification algorithms.

Figure 12. F-Score in WPDP_C1

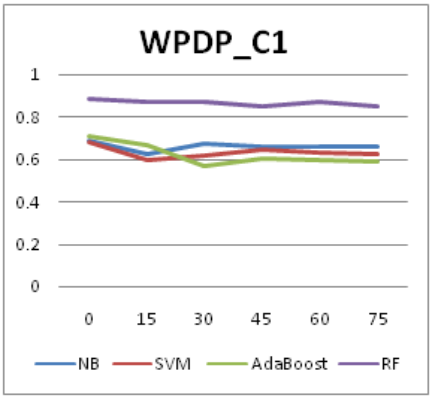


Figure 13. F-Score in WPDP_C2

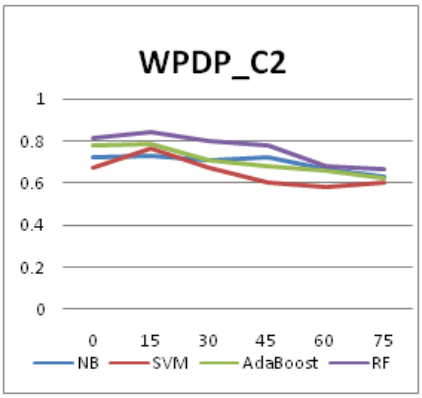


Figure 14. F-Score in WPDP_C3

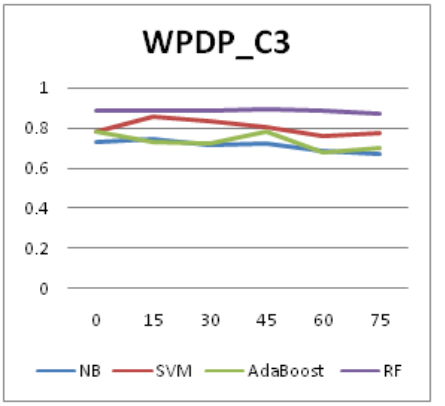
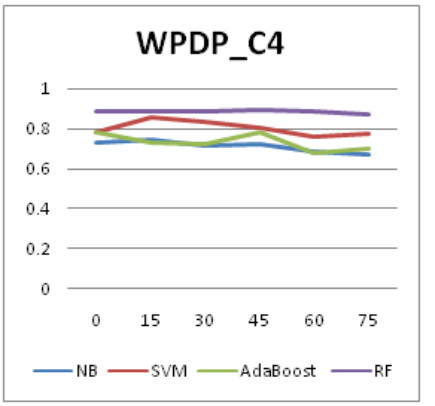


Figure 15. F-Score in WPDP_C4



RQ2. To compare the prediction performance of HCPDP at various levels of noise with or without handling of CIP.

SRCM is used to find strongly correlated feature pairs for all six heterogeneous prediction combinations after completing data preprocessing and feature engineering on datasets. Table 7 shows how many correlated feature pairs there are in each combination with a CCV larger than 0.05. (cutoff threshold). According to the state of the art, the cutoff is set at 0.05 to encompass the maximum defect prediction likelihood in all feasible dataset combinations. As per Table 7, the source and target datasets JDT and ar1 have a total of 19 strongly correlated feature pairs.

Table 7. Metric matching in HCPDP

Heterogeneous Prediction Combination	No. of Strongly Correlated Feature Pairs
HCPDP_C1	19
HCPDP_C2	17
HCPDP_C3	12
HCPDP_C4	19
HCPDP_C5	11
HCPDP_C6	17

The FPR values for defect prediction in the target project utilizing a heterogeneous source project are shown in Table 8. After using CBA to manage an unbalanced dataset, the maximum and minimum FPR values for the heterogeneous prediction combination HCPDP_C1 are determined to be 0.60 for SVM and 0.11 for RF, respectively. Similarly, the highest and lowest FPR values for HCPDP_C4 are 0.67 using AdaBoost and 0.16 using RF. It can be seen that the model in HCPDP_C5 and HCPDP_C6 tries to forecast defects in the target dataset ar3 using different source datasets EQ and LC.

Table 8. FPR values at different level of noise (HCPDP)

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
HCPDP-C1	NB	0.32/0.38	0.34/0.40	0.31/0.43	0.31/0.48	0.35/0.42	0.41/0.50
	SVM	0.59/0.67	0.60/0.60	0.51/0.58	0.47/0.55	0.41/0.49	0.34/0.46
	AdaBoost	0.43/0.62	0.44/0.52	0.41/0.50	0.49/0.52	0.43/0.48	0.62/0.71
	RF	0.28/0.49	0.16/0.46	0.11/0.40	0.12/0.36	0.35/0.39	0.35/0.36
HCPDP-C2	NB	0.31/0.49	0.20/0.26	0.26/0.39	0.21/0.36	0.34/0.29	0.31/0.35
	SVM	0.68/0.74	0.68/0.77	0.51/0.70	0.41/0.68	0.49/0.55	0.51/0.67
	AdaBoost	0.42/0.56	0.55/0.59	0.50/0.65	0.42/0.69	0.40/0.58	0.59/0.71
	RF	0.27/0.35	0.25/0.32	0.27/0.45	0.18/0.31	0.19/0.25	0.29/0.37
HCPDP-C3	NB	0.30/0.41	0.33/0.37	0.41/0.46	0.43/0.52	0.51/0.55	0.61/0.65
	SVM	0.32/0.41	0.37/0.42	0.40/0.51	0.44/0.50	0.53/0.57	0.68/0.62
	AdaBoost	0.20/0.26	0.21/0.29	0.27/0.30	0.38/0.51	0.49/0.58	0.55/0.60
	RF	0.19/0.27	0.16/0.32	0.15/0.38	0.27/0.39	0.38/0.43	0.37/0.41
HCPDP-C4	NB	0.21/0.29	0.21/0.37	0.32/0.36	0.21/0.37	0.38/0.49	0.40/0.47
	SVM	0.28/0.34	0.19/0.25	0.23/0.35	0.21/0.34	0.43/0.56	0.49/0.57
	AdaBoost	0.21/0.34	0.29/0.31	0.37/0.46	0.48/0.59	0.41/0.59	0.67/0.71
	RF	0.21/0.29	0.16/0.23	0.24/0.29	0.31/0.46	0.38/0.44	0.59/0.61

Table 8 continued on next page

Table 8 continued

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
HCPDP-C5	NB	0.22/0.47	0.34/0.37	0.34/0.39	0.35/0.36	0.37/0.40	0.35/0.38
	SVM	0.78/0.89	0.67/0.77	0.59/0.67	0.44/0.58	0.39/0.49	0.31/0.44
	AdaBoost	0.79/0.78	0.5/0.58	0.53/0.64	0.67/0.48	0.47/0.52	0.40/0.40
	RF	0.24/0.32	0.26/0.59	0.18/0.44	0.18/0.37	0.22/0.35	0.25/0.44
HCPDP-C6	NB	0.18/0.43	0.13/0.37	0.30/0.36	0.33/0.34	0.30/0.42	0.31/0.33
	SVM	0.66/0.78	0.67/0.71	0.53/0.60	0.39/0.51	0.33/0.44	0.25/0.37
	AdaBoost	0.40/0.73	0.48/0.55	0.55/0.67	0.57/0.55	0.44/0.49	0.34/0.37
	RF	0.21/0.27	0.20/0.41	0.18/0.34	0.18/0.31	0.13/0.32	0.12/0.39

Table 9. TPR values at different level of noise (HCPDP)

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
HCPDP-C1	NB	0.50/0.59	0.51/0.64	0.48/0.61	0.55/0.68	0.44/0.49	0.31/0.69
	SVM	0.66/0.71	0.69/0.64	0.61/0.64	0.58/0.60	0.50/0.57	0.48/0.51
	AdaBoost	0.68/0.61	0.68/0.65	0.62/0.66	0.58/0.73	0.51/0.70	0.34/0.48
	RF	0.80/0.71	0.82/0.68	0.80/0.75	0.86/0.72	0.66/0.57	0.57/0.52
HCPDP-C2	NB	0.66/0.56	0.69/0.56	0.76/0.46	0.70/0.57	0.55/0.48	0.49/0.55
	SVM	0.65/0.54	0.77/0.63	0.80/0.63	0.69/0.53	0.58/0.48	0.60/0.54
	AdaBoost	0.72/0.66	0.70/0.62	0.79/0.56	0.62/0.55	0.61/0.50	0.55/0.43
	RF	0.90/0.74	0.92/0.76	0.89/0.70	0.90/0.70	0.69/0.47	0.60/0.39
HCPDP-C3	NB	0.80/0.72	0.84/0.80	0.79/0.61	0.80/0.68	0.60/0.51	0.58/0.45
	SVM	0.82/0.66	0.86/0.73	0.77/0.70	0.64/0.43	0.66/0.54	0.41/0.38
	AdaBoost	0.85/0.71	0.85/0.77	0.73/0.58	0.70/0.58	0.68/0.50	0.50/0.44
	RF	0.94/0.84	0.90/0.87	0.90/0.82	0.84/0.61	0.70/0.62	0.55/0.32
HCPDP-C4	NB	0.67/0.60	0.72/0.61	0.76/0.60	0.76/0.69	0.56/0.44	0.43/0.33
	SVM	0.78/0.68	0.74/0.60	0.73/0.54	0.68/0.50	0.61/0.46	0.53/0.38
	AdaBoost	0.83/0.70	0.85/0.64	0.71/0.60	0.44/0.40	0.37/0.21	0.35/0.25
	RF	0.90/0.72	0.86/0.71	0.90/0.74	0.88/0.64	0.72/0.54	0.68/0.53

Table 9 continued on next page

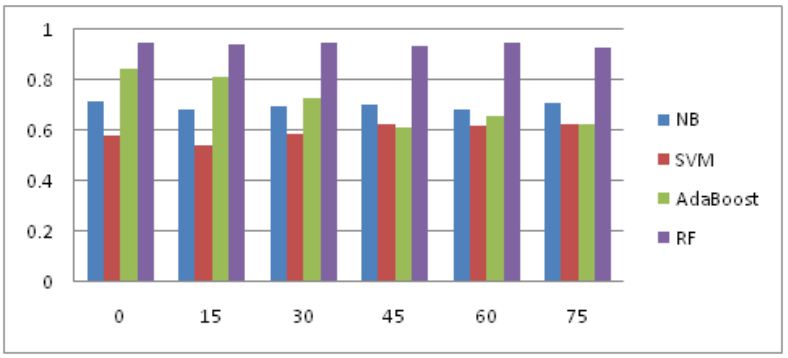
Table 9 continued

Prediction Combination	Classification Algorithm	Induced Noise Level (%)					
		0	15	30	45	60	75
HCPDP-C5	NB	0.73/0.71	0.74/0.68	0.71/0.68	0.72/0.70	0.68/0.69	0.67/0.62
	SVM	0.77/0.72	0.85/0.72	0.83/0.69	0.79/0.68	0.75/0.67	0.77/0.66
	AdaBoost	0.78/0.75	0.72/0.71	0.78/0.72	0.67/0.71	0.70/0.68	0.66/0.64
	RF	0.89/0.78	0.89/0.76	0.88/0.74	0.89/0.76	0.88/0.74	0.88/0.72
HCPDP-C6	NB	091/0.92	0.86/0.83	0.78/0.73	0.73/0.61	0.63/0.58	0.65/0.54
	SVM	0.91/0.85	0.84/0.72	0.79/0.58	0.68/0.51	0.62/0.43	0.66/0.48
	AdaBoost	0.95/0.87	0.86/0.85	0.81/0.74	0.67/0.59	0.48/0.49	0.48/0.46
	RF	0.97/0.90	0.94/0.81	0.92/0.68	0.87/0.65	0.86/0.65	0.84/0.61

The FPR values in HCPDP_C5 and HCPDP_C6 at the highest level of induced noise are 0.25/0.44 and 0.12/0.39, respectively, according to the results of Table 8. When compared to HCPDP_C5, the better outcomes for every classification algorithm in HCPDP_C6 are due to the higher number of strongly correlated pairs produced by the metric matching phase. This demonstrates that, in the case of HCPDP, metric matching has a significant impact on prediction performance. When the DP performance is evaluated using TPR values, similar performance patterns can be seen in the results of Table 9. It shows that the maximum and minimum TPR values for HCPDP_C4 after including CBA to handle CIP in training dataset Safe with CIR as 64.71 are 0.90 using RF and 0.35 using AdaBoost, respectively.

Using a 10-fold cross validation technique, the average prediction accuracies for all HCPDP combinations are examined. At all levels of induced noise, RF outperforms the other set of employed classifiers, as shown in Figure 16.

Figure 16. HCPDP's prediction accuracy



Tables 10 and 11 exhibit the comparative results for WPDP and HCPDP prediction performance. The authors used the prediction combinations WPDP_C1 to WPDP_C4 and HCPDP_C1 to HCPDP_

C4 for this study. While the source and target projects are the same in With-in predictions, they are different in heterogeneous prediction, as seen in Tables 10 and 11. For all four prediction combinations, it can be seen that HCPDP performs similarly to WPDP. Table 10 shows that after addressing CIP with CBA, the FPR values for WPDP_C1 and HCPDP_C1 are 0.313 and 0.375, respectively, at a noise level of 75 percent.

Table 10. Comparison of prediction performance of WPDP & HCPDP using FPR

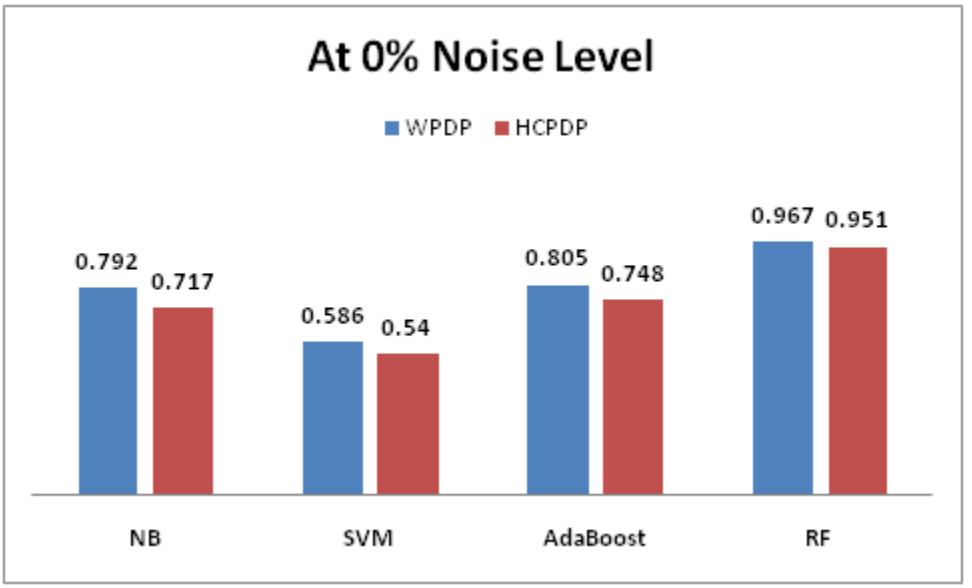
Prediction Combination	Induced Noise Level (%)					
	0	15	30	45	60	75
(ar1, ar1)	0.395	0.408	0.363	0.343	0.350	0.313
(JDT, ar1)	0.405	0.410	0.335	0.397	0.385	0.375
(JDT, JDT)	0.443	0.400	0.405	0.320	0.315	0.283
(ar5, JDT)	0.420	0.420	0.410	0.305	0.330	0.425
(Apache, Apache)	0.268	0.283	0.345	0.428	0.465	0.495
(ar1, Apache)	0.253	0.308	0.308	0.380	0.478	0.553
(ML, ML)	0.205	0.203	0.225	0.265	0.318	0.323
(Safe, ML)	0.228	0.213	0.290	0.303	0.400	0.538

When there is no induced noise, the maximum TPR values for predicting defects in target dataset Apache using source datasets Apache (With-in) and ar1 (HCPDP) are 0.908 and 0.853, respectively, according to Table 11. After treating CIP with CBA, all experimental results of Tables 10 &11 for this comparison analysis are gathered. As shown in Figure 17, the DP performance within a project surpasses predictions using heterogeneous projects at zero level of generated noise in terms of average AUC value. On the basis of these findings, one can conclude that HCPDP performance is comparable to defect prediction within the project.

Table 11. Comparison of prediction performance of WPDP & HCPDP using TPR

Prediction Combination	Induced Noise Level (%)					
	0	15	30	45	60	75
(ar1, ar1)	0.690	0.670	0.653	0.640	0.660	0.593
(JDT, ar1)	0.660	0.675	0.628	0.643	0.528	0.450
(JDT, JDT)	0.790	0.755	0.755	0.708	0.705	0.673
(ar5, JDT)	0.733	0.770	0.810	0.728	0.608	0.560
(Apache, Apache)	0.908	0.870	0.798	0.740	0.678	0.648
(ar1, Apache)	0.853	0.863	0.798	0.745	0.660	0.510
(ML, ML)	0.758	0.718	0.713	0.703	0.708	0.688
(Safe, ML)	0.795	0.793	0.700	0.690	0.565	0.498

Figure 17. Comparative analysis of WPDP & HCPDP using AUC value



RQ3. To determine the maximum level of noise that each prediction pair can tolerate under both SDP categories (HCPDP & WPDP).

For both SDP categories, the range of admissible noise varies depending on the classification algorithm as shown in Tables 12 & 13. As shown in Table 12, the performance of WPDP_C1 (AdaBoost), WPDP_C3 (NB), and WPDP_C4 (NB) in predicting defects within a project is uniform. This indicates that there is no abnormally high or low point in either FPR or TPR values throughout the experimented noise range. However, using the AdaBoost classification algorithm, the values of FPR and TPR for WPDP_C3 are found to be similar or show slight fluctuation in their values between 15% and 45%. On the other hand, WPDP_C4 performed consistently from pure data (at 0% noise level) through 45% noise level, with a breakdown point beyond 45% noise level. All six heterogeneous prediction pairs showed similar kind of tolerated noise ranges like WPDP. Except for SVM, HCPDP_C5 provides consistent prediction performance for all classification techniques. In the case of SVM, no specific performance pattern can be identified across the entire noise range.

Table 12. Tolerable range of noise in With-In prediction combinations

Prediction Combinations	Classifier	Tolerable Noise Range	
		Lower	Upper
WPDP_C1	NB	0	60
	SVM	0	15
	AdaBoost	Perform Uniformly	
	RF	0	45

Table 12 continued on next page

Table 12 continued

Prediction Combinations	Classifier	Tolerable Noise Range	
		Lower	Upper
WPDP_C2	NB	Perform Uniformly	
	SVM	0	30
	AdaBoost	0	30
	RF	0	30
WPDP_C3	NB	0	15
	SVM	0	15
	AdaBoost	15	45
	RF	0	30
WPDP_C4	NB	Perform Uniformly	
	SVM	15	60
	AdaBoost	0	30
	RF	0	45

Table 13. Tolerable range of noise in heterogeneous prediction combinations

Prediction Combinations	Classifier	Tolerable Noise Range	
		Lower	Upper
HCPDP_C1	NB	0	60
	SVM	0	45
	AdaBoost	0	60
	RF	15	45
HCPDP_C2	NB	15	45
	SVM	0	30
	AdaBoost	Perform Uniformly	
	RF	0	45
HCPDP_C3	NB	15	45
	SVM	0	30
	AdaBoost	0	30
	RF	15	45
HCPDP_C4	NB	15	45
	SVM	0	45
	AdaBoost	0	15
	RF	0	30

Table 13 continued on next page

Table 13 continued

Prediction Combinations	Classifier	Tolerable Noise Range	
		Lower	Upper
HCPDP_C5	NB	Perform Uniformly	
	SVM	No Proper Performance Pattern	
	AdaBoost	Perform Uniformly	
	RF	Perform Uniformly	
HCPDP_C6	NB	0	15
	SVM	0	30
	AdaBoost	0	30
	RF	Perform Uniformly	

RQ4. Which classification method outperforms the rest of the algorithms in use?

From Figures 11 and 15, it is clear that RF outperforms all other classifiers. In both WPDP and HCPDP, the prediction accuracies using RF are higher for all prediction combinations. In comparison to other classifiers, the AUC plot indicated that RF performed consistently and is least impacted by noise. So, the increasing order of DP performance is given as follows: -

SVM < AdaBoost < NB < RF

CONCLUSION & FUTURE DIRECTIONS

HCPDP is an open study area in software defect prediction that forecasts defects in the target application without using previous defect data. The two major problems of SDP, noise and an unbalanced dataset, were investigated in this research employing 133 experiments on three open-source projects with ten prediction combinations. CIP has been found to have a substantial impact on any DP model and should be dealt with during the pre-processing step or before to train the DP model. In this study, a novel hybrid technique called CBA is developed to overcome the drawbacks of both oversampling and undersampling data-driven CIL approaches. The results of the experiment were analyzed using four different classifiers, and it was discovered that RF performed the best among all classifiers and was less affected by noise level.

It is also found that WPDP and HCPDP perform similarly across a noise range, but WPDP outperforms HCPDP for pure data since it is evident that a DP model trained using the same project data will perform better than a model trained using defect data from different projects.

HCPDP modeling relies heavily on metric matching. Poor performance will always be the outcome of ineffective metric matching between prediction datasets. As a future project, a novel technique for preventing noisy metric matching could be investigated. Deep learning techniques can be used to investigate new features from the source and target projects in order to improve the HCPDP model's performance. Future research should focus on developing an empirical association between software defect prediction and predictive maintenance.

REFERENCES

Bettenburg, N., Hassan, A. E., & Nagappan, M. (2012). Think locally, act globally: Improving defect and effort prediction models. In *9th IEEE Working Conference on Mining Software Repositories (MSR)* (pp. 60–69). IEEE. doi:10.1109/MSR.2012.6224300

Briand, L. C., Melo, W. L., & Wurst, J. (2002). Assessing the applicability of fault- proneness models across object-oriented software projects. *IEEE Transactions on Software Engineering*, *2*(8), 706–720. doi:10.1109/TSE.2002.1019484

Canfora, G., De Lucia, A., Oliveto, R., Panichella, A., Di Penta, M., & Panichella, S. (2013). Multi-objective cross-project defect prediction. In *IEEE Sixth International Conference on Verification and Validation in Software Testing*. IEEE.

Cruz, C., & Ochimizu, A. E. (2009). 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)*, 460-463.

D'Ambros, M., Lanza, M., & Robbes, R. (2012). Evaluating defect prediction approaches: A benchmark and an extensive comparison. *Empirical Software Engineering*, *17*(4-5), 531–577. doi:10.1007/s10664-011-9173-9

Diwaker, C., Tomar, P., Solanki, A., Nayyar, A., Jhanjhi, N. Z., Abdullah, A., & Supramaniam, M. (2019). A new model for predicting component-based software reliability using soft computing. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 147191–147203.

Gao, K., Khoshgoftaar, T. M., Zhang, H., & Seliya, N. (2011). Choosing software metrics for defect prediction: An investigation on feature selection techniques. *Software, Practice & Experience*, *41*(5), 579–606.

Gheisari, M., Panwar, D., Tomar, P., Harsh, H., Zhang, X., Solanki, A., & Alzubi, J. A. (2019). An optimization model for software quality prediction with case study analysis using MATLAB. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 85123–85138.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Jiang, K., Zhang, Y., Wu, H., Wang, A., & Iwahori, Y. (2020). Heterogeneous Defect Prediction Based on Transfer Learning to Handle Extreme Imbalance. *Appl. Sci.* 10.3390/app10010396

Jing, X., Dong, X., Qi, F., Wu, F., & Xu, B. (2015). Heterogeneous cross company defect prediction by unified metric representation and CCA- based transfer learning. In *Proceedings of the 2015 in 10th Joint Meeting on Foundations of Software Engineering* (pp. 496–507). ACM. doi:10.1145/2786805.2786813

Kukkar, A., Mohana, R., Kumar, Y., Nayyar, A., Bilal, M., & Kwak, K. S. (2020). Duplicate bug report detection and classification system based on deep learning technique. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 200749–200763.

Lee, S. P., & Felix, E. A. (2020). Predicting the number of defects in a new software version. *PLoS One*, *15*(3).

Li, B., He, P., & Ma, Y. (2014). Towards cross-project defect prediction with imbalanced feature sets. CoRR, vol.abs/1411.4228.

Linberg, K. R. (1999). Software developer perceptions about software project failure: A case study. *Journal of Systems and Software*, *49*(2-3), 177–192. doi:10.1016/S0164-1212(99)00094-1

Majd, A., Vahidi-Asl, M., Khalilian, A., Poorsarvi-Tehrani, P., & Haghighi, H. (2020). SLDeep: Statement-level software defect prediction using deep-learning model on static code features. *Expert Systems with Applications*, *14*(7).

Marqués, A., García, V., & Sánchez, J. (2013). On the suitability of resampling techniques for the class imbalance problem in credit scoring. *The Journal of the Operational Research Society*, *64*, 1060–1070. doi:10.1057/jors.2012.120

Menzies, T., Butcher, A., Cok, D. R., Marcus, A., & Zimmermann, T. (2011). Local vs. global models for effort estimation and defect prediction. In *26th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 343–351). IEEE. doi:10.1109/ASE.2011.6100072

Mwadulo, M. W. (2015). A Review on Feature Selection Methods for Classification Tasks. *International Journal of Computer Applications Technology and Research*, *5*(6), 395–402. doi:10.7753/IJCATR0506.1013

Ni, C., Liu, W., Gu, Q., Chen, X., & Chen, D. (2017). FeSCH: A Feature Selection Method using Clusters of Hybrid-data for Cross-Project Defect Prediction. *Proceedings of the 41st IEEE Annual Computer Software and Applications Conference, COMPSAC*, 51–56.

Rahman, F., Devanbu, P., & Posnett, D. (2012). 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)* (pp. 61-65). ACM.

Turhan, B., Menzies, B., Bener, A. B., & Stefano, J. (2009). On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, *14*(5), 540–578. doi:10.1007/s10664-008-9103-7

Vashisht, R., & Rizvi, S. M. (2020). Feature Extraction to Heterogeneous Cross Project Defect Prediction. *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO),* 1221-1225. DOI: doi:10.1109/ICRITO48877.2020.9197799

Vashisht, R., & Rizvi, S. M. (2021). Feature Engineering to Heterogeneous Cross Project Defect Prediction- A Novel Framework. *Arabian Journal for Science and Engineering*.

Rohit Vashisht is an Assistant Professor in the Department of Computer Science & Information Technology, KIET Group of Institutions, Delhi-NCR, Ghaziabad, India. He has completed his B.Tech from Ajay Kumar Garg Engineering College, AKTU with silver medal and M.Tech from USICT, GGSIPU, and Delhi with gold medal. He is currently pursuing Ph.D from Jamia Millia Islamia University. His area of interest includes machine learning, sentimental analysis, software engineering and cloud computing. He has teaching experience of 5.5 years. He is certified as Elite Silver for subject Compiler Design and Operating System by NPTEL. He has good numbers of publications in reputed international conferences and journals (indexed in Scopus, ESCI, SCIE).

S. A. M. Rizvi has been working as a professor for a decade in the Department of Computer Science, Jamia Millia Islamia, having more than 35 years of experience in teaching and research at Universities/HEIs in India and abroad. He has earned doctorate in Computer Science way back in 1996from Dr. R. M. L. Avadh University, India. He is completing two decades of Teaching and Research at Central University - Jamia Millia Islamia where he was appointed Head of the Department for 3-years term from 2016 till 2019. He designed various programmes/courses as a Chairman/Member of BOS, Academic Council, and other academic bodies at universities/ Higher Educational Institutions (HEIs). He has more than 187+ Publications as per Google Scholar, including SCI, SCOPUS and IEEE Transactions covering a vast array of topics in Computer Science and Applications. He has to his credit More than 23 Ph.D. awardees till date, with 8 currently registered scholars. He is an author of six text books in the subject well known and respected amongst peers and Scholars. He is a Senior Member of the Computer Society of India (CSI), Old Member of IEEE, ISCA, and IEA. Prof Rizvi is a versatile personality having taught, and held Senior Academic Positions across India, such as Goa University, University in Chennai, in Haryana, at Agra University in U.P. and abroad as well. He taught in the USA (Credit Hour System), Australia, UAE, and Indian Educational Systems and has the exposure of working with International Accreditation Bodies and getting approvals for new programmes launched, such as B.S.(MIS) of AbuDhabhi University. He has also worked as Director, Training and University-Industrial Linkage programme and Chief Manager (EDP/IT) in Goa Shipyard under the Ministry of Defense, Government of India. He has Trained Industrial Computer Professional in the area of software development and has hands-on experience through various Training of Executive-MBA programmes. He has been a Founder-Director and Head at various Universities/HEIs where he has innovated new courses and programmes. Prof Rizvi is also been a part of various Selection Committees at Central and State Universities, State PSCs, including JKPSC, Recruitments/Promotions at Nationalized Bank besides President of RWAs. His research interests in Genome/Bioinformatics led him to establish a vast lab in the shape of FARM on seeds, Medicinal plant, developed like a Botanical Garden. These varieties of interests and entrepreneurial skills made him all the more popular amongst Academia. He observes the research world from his perspective.