Arabian Journal for Science and Engineering (2023) 48:2539–2560
https://doi.org/10.1007/s13369-022-07337-9

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Feature Engineering to Heterogeneous Cross Software Projects Defect Prediction: A Novel Framework

Rohit Vashisht[1,2] · **Syed Afzal Murtaza Rizvi**[1]

Received: 22 May 2021 / Accepted: 7 June 2022 / Published online: 2 November 2022
© King Fahd University of Petroleum & Minerals 2022

Abstract
Heterogeneous cross-project defect prediction (HCPDP) aims to predict defects in a target project with limited historical defect data via a defect prediction (DP) model trained with defect data of another source project. The accuracy of a DP model is highly dependent on the set of features selected in feature engineering (FE) phase. The study evaluates the effectiveness of proposed four-phase HCPDP framework with more focus on FE phase using the stacking-based ensemble learning method. Auto-encoder (AE), a deep learning-based FE technique is used for the proposed analysis. In addition, two novel techniques to deal with imbalance dataset and to determine correlation between features are also proposed in this paper. For comparative analysis, accuracy, recall, F-score and area under curve (AUC) are used as the output parameters. To compare DP model's output with or without FE phase, ten prediction pairs from four open source projects have been considered. The experimental results show that the AE technique is able to reduce the number of features by an average of 50% as compared to data-driven approaches. Also, the proposed model gave better performance in comparison with traditional heterogeneous models with highest AUC of 0.8901.

Keywords Feature Engineering · Cross-Project · Classification · Heterogeneous · Ensemble · Cross-Validation

1 Introduction

Software has become an indispensable part of every human's day-to-day activities. In today's scenario, many important fields such as education, marketing, banking and transport need highly reliable, defect-free and high-quality software applications, as any failure in these applications can result in enormous losses from finance to human lives. Software errors may be due to inconsistencies, ambiguities, oversights or misinterpretation of the specifications to be met by the software, carelessness or negligence in writing code, insufficient testing, inappropriate or unexpected use of the software or other unforeseen issues. In order to reduce the significant cost of software development, it is very important to identify these software defects at the right time. "Software testing should be done for early identification of software faults because amendments in maintenance phase will lead to huge cost that grows exponentially if faults are identified in later stages of Software Development Life Cycle (SDLC)", as described in [1]. In comparison, the SDLC's software testing phase absorbs 60% of the overall cost of software development. Therefore, it is very critical that testing on the right modules should be performed at the right time.

According to the state of the art, software defect prediction (SDP) can be broadly divided into two groups- with-in project defect prediction (WPDP) and cross-project defect prediction (CPDP). In WPDP, the available defect dataset is split up into two parts in order to build the DP model in such a way that the DP model is trained using one part of the dataset (referred to as labeled observations) and the other part is used to validate DP model as shown in Fig. 1. Testing the DP model involves finding labels that are either faulty or non-faulty for unidentifiable instances in the target dataset [2].

According to the state of the art, software defect prediction (SDP) can be broadly divided into two groups- with-in project defect prediction (WPDP) and cross-project defect prediction (CPDP). In WPDP, the available defect dataset is split up

✉ Rohit Vashisht
rohitvashisht794@gmail.com

Syed Afzal Murtaza Rizvi
sarizvi@jmi.ac.in

[1] Department of Computer Science, Jamia Millia Islamia, Delhi, India

[2] Department of Computer Science & Information Technology, KIET Group of Institutions, Delhi-NCR, Ghaziabad, India

into two parts in order to build the DP model in such a way that the DP model is trained using one part of the dataset (referred to as labeled observations) and the other part is used to validate DP model as shown in Fig. 1. Testing the DP model involves finding labels that are either faulty or non-faulty for unidentifiable instances in the target dataset [2].

CPDP is another class of SDP in which software projects that lack the needed local defect data can use data from other projects to construct an accurate and effective DP model. In addition, CPDP can be further categorized into homogeneous CPDP (HoCPDP) and heterogeneous CPDP (HCPDP) subcategories. The common software measures/features are collected by HoCPDP from both the source application (whose defect data is employed to train the SDP model) and the target application (for which the SDP model is made) [3]. But, there are no uniform metrics between the prediction pair datasets when using HCPDP. Through measuring the coefficient of correlation between all feasible software feature combinations, uniform features may be found between two applications. In order to forecast project-wide defects, the combinations of feature pairs displaying some sort of analogous distribution in their values are used as common features between source and target datasets in case of HCPDP. For example, (A,Q), (B,P) and (D,S) are correlated feature pairs for HCPDP category as depicted in Fig. 2. More information about both CPDP categories is shown in Fig. 2.

Irrelevant or useless software features chosen during the FE step can be one of the key explanations for less accurate DP model as redundant collection of features can lead to skewed or misleading prediction performance. Therefore, the important issue that should be tackled first in order to build a highly reliable SDP model is the selection of the right set of features from a given pool of input features. The article studies tests the prediction performance of three-phased WPDP model and four-phased HCPDP-AE model with or without FE phase using different classification algorithms.

Data-driven FE techniques such as principal component analysis (PCA) can only model linear relationships among input features. In contrast to data-based FE models, neural nets can model nonlinear transformations of features and perform better as the number of features grows.

Auto-encoder (AE), an unsupervised artificial neural network (ANN) and encoding–decoding-based FE technique, has been used in the proposed research study to map higher-dimensional feature data to lower one along with elimination of redundant and noisy features. The motivation behind this study is to evaluate the prediction performance of the four-stage HCPDP-AE model with introduction of the novel approach to implement each stage with a greater emphasis on the FE stage. In addition, two novel techniques to deal with imbalance dataset and to determine correlation among features in HCPDP are also proposed in the paper. Both techniques overcome the shortcomings of traditional methods in

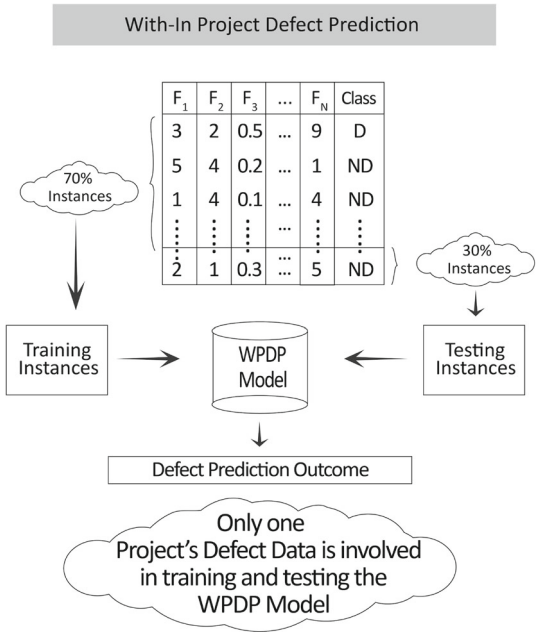


Fig. 1 With-in project defect prediction

their domain as described in later section. The main contribution areas of the study are as follows:-

RQ1. Compare and contrast the use of data-driven and deep learning-based FE strategies with the conventional approach of DP, i.e. WPDP.

RQ2. Compare the prediction performance of proposed HCPDP framework with or without FE phase.

RQ3. Whether and to what extent DP results of HCPDP's model are comparable to the outcomes of WPDP's model?

RQ4. Compare and validate the performance of the proposed HCPDP framework with existing benchmarked heterogeneous prediction models.

The outline of the paper is as follows:—Section 2 includes a comprehensive analysis of HCPDP's related work, Sect. 3 describes the four-phase HCPDP-AE model and the three-phase WPDP model with detailed explanation of each phase, Sect. 4 describes the datasets used to execute the proposed work and the output metrics used to assess the experimental results, the development part of the experiments is explained in Sect. 5, the experimental outcomes are discussed in Sects. 6 and 7 explains the threats to construct validity and the conclusive findings are outlined in Sect. 8.

2 Related Work

In 2002, Melo et al. [4] reported the first known study in CPDP. They introduced the MARS (Multivariate Adaptive Regression Spline) paradigm for defect prediction and data architecture in two Java-based frameworks, Xpose and

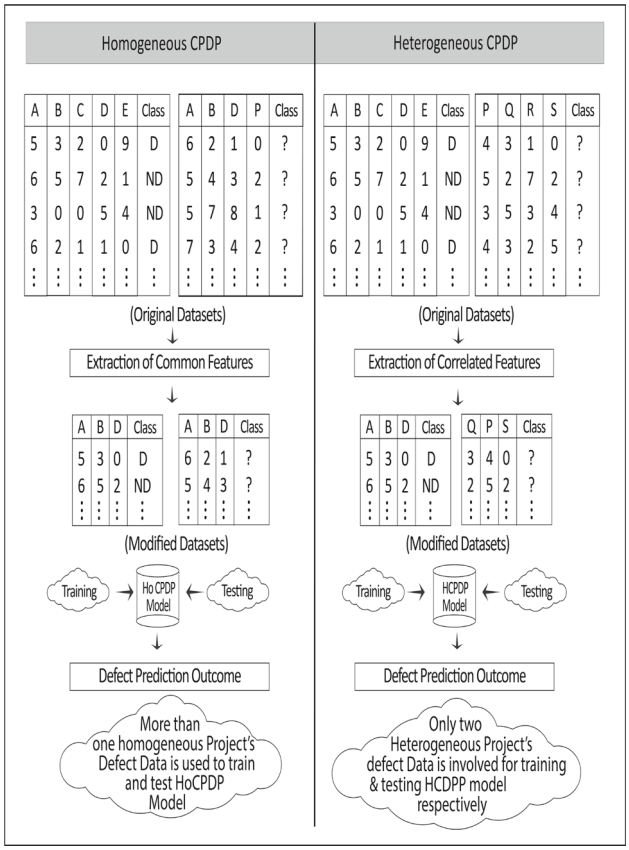


Fig. 2 Categories of cross software projects defect prediction

Jwriter. They used their proclivity for fault to forecast the classes in Jwriter. They did this by using a model trained on the Xpose dataset. They compared MARS' efficiency to that of linear regression (LR) and discovered that MARS outperforms LR and is much more cost-effective.

Menzies et al. [5] used data from ten projects from two different sources in 2009. They filter down the data for successful defect prediction by eliminating noisy, repetitive and irrelevant data and train the model with this unblended data. The tests were carried out using the nearest neighbor (NN) method on data from ten projects. The findings showed that the tests were effective at predicting defects within a project. Meanwhile, using these experiments, the CPDP task was unable to outperform the project defect prediction task.

In same year 2009, Camargo et al. [6] used log transformation for the first time to identify related instances in training and analyzing project data to eliminate project-based data instances. The classification for defect prediction on Internet Explorer and Mozilla Firefox as training and testing projects was suggested by Menzies et al. [5] in the same year. For the classification task, they used the coding model and process parameters. They used Mozilla Firefox defect data to train the proposed DP model, which was then used to predict defects in Internet Explorer. These experiments revealed that when the proposed model was used as a training project and Mozilla Firefox was used as a testing project, the proposed model outperformed.

Menzies et al. [7] argued in 2011 that relevancy varies depending on interpretation. They said that relevance differs with interpretation, and that data relevance can be contradictory depending on how it is perceived. When viewed globally, data that seems to be significant can be meaningless when viewed locally. They supported their claims with experiments, concluding that local behavior was superior to global behavior and that condition-based laws should be prioritized over taking into account other factors.

In 2012, Bellenburg et al. [8] added to Menzies et al. [7] arguments by demonstrating that local models were better for a specific dataset, but global models were better for generality. In the same year, Rahman et al. [9] conducted studies to show that performance indicators such as F-score, accuracy and recall are not sufficient for quality assurance when defect prediction is made using various models. They said that AUC gives the comparable results in WPDP models.

To address the shortcomings of single objective model [9], in 2013, Canfora et al. [10] suggested a multi-objective approach. They used a non-dominated sorted generic algorithm (NSGA-II) to practice the logistic regression (LR) model.

In 2011, Gao et al. [11] developed a universal defect prediction (UDP) model using 1398 projects from Google code and source forge. This model compares the metrics in the training and testing projects' datasets, and if at least 26 of them fit, then only predictions could be made on target project. He et al. [3] overcame this constraint by developing a new metric based on instance characteristic vectors. They also found unfavorable effects when comparing CPDP to feature disparity. The tests were carried out on 11 projects using three different datasets.

In 2014, He et al. [3] compared the output findings for WPDP and CPDP using feature selection approaches. They discovered that the lower the number of training project features used to train classifiers, the higher the precision in WPDP and the greater the F-score and recall performance in CPDP. Various ensemble classifiers are also trained and validated for the CPDP task [12, 13].

Dong et al. [14] suggested canonical correlation analysis (CCA) approach to model defects. They became the first to put the study for heterogeneous defect prediction (HDP) to the public's attention. By supplementing dummy metrics with null values, they eradicate the metrics disparity issue between the training and testing project datasets. They tested 14 projects using four different datasets.

In 2015, Fu et al. [15] performed studies on 34 projects with 5 datasets. They suggested HDP task using transfer learning system. They do not use null values to augment

metrics as Dong et al. [14] suggested, but their findings are equivalent to WPDP.

Ryu, Jang and Baik [16] used a new approach called the transfer cost-sensitive boosting method to execute the CPDP challenge in the same year. For the CPDP task, their approach generated state-of-the-art performance. They also suggested a CPDP challenge that takes into account class-imbalance using a multi-objective Naive Bayes technique (MONBT) [17]. All WPDP models, as well as single objective models, were outperformed by their MONBT.

Jing et al. [18] created a novel unified metric representation (UMR) for predicting heterogeneous defects in 2015. Fu et al. [15] suggested an HDP challenge based on metric collection and metric matching in the same year. They studied 28 projects and found that the proposed approach was superior to WPDP and, in some circumstances, outperformed it statistically.

Ni et al. [19] proposed FESCH in 2017, a novel approach that outperformed both TCA+ and WPDP in most scenarios and gave state-of-the-art results for the baseline methods used. Furthermore, the findings indicated that FeSCH's success was self-sustaining and independent of the classifiers used.

In same year, Li et al. [20] compared the four filtration approaches for defect data. They said that the fault data filtration approach chosen has a significant impact on the model's ability to predict defects. They compared four different filtering techniques: data characteristic-based filter (DCBF), target project data-guided filter (TGF), source project data-guided filter (SGF) and local cluster-based filter (LCBF). They also introduced a new filter, the hierarchical selection-dependent filter (HSDF), to resolve the shortcomings of the previous four filters in terms of scalability when dealing with massive datasets. The proposed filtering strategy outperformed existing filtering techniques.

Xu et al. [21] proposed a domain adaptation approach for reducing the higher-dimensional features of training and analyzing project domains in 2018. To learn the difference between function spaces, they used the dictionary learning technique. They compared heterogeneous defect adaptation (HDA) [21], CCA+ [14] and HDP [15] using three open source projects: NetGene, NASA and AEEEM, and three performance measures: recall, F-score and balance.

In 2020, Lee and Felix [22] concentrated on method-level (ML) defect estimation using regression models in a recent software release after gathering-related data from previous versions of the same system. The authors used three performance measurement variables, such as defect density, defect velocity, and time to implement defects that show a significant relationship with ML defects. The proposed work also facilitated the study and evaluation of pre-to-post system data preprocessing classifiers and entropy values in average output datasets. The defect velocity had the highest correlation with the count among ML faults among all three factors, with a 93% correlation.

Majd et al. [23] suggested using deep learning models to forecast statement class defects (SCD) in same year. The authors of this paper sought to relieve the burden on software developers by defining places or modules that are more vulnerable to defects. The authors used Code4Bench's Broad Short-Term Memory (BSTM) deep learning model to run experiments on 1,19,989 C/C++ programs. The authors have put the SCD model to the test for predicting defects in unseen data (i.e., new statements) and found that it performed well, with high memory, precision and accuracy.

In the field of HCPDP, Grassmann manifold optimal transfer defect prediction (GMOTDP) is a recent novel work in year 2020. Jiang et al. [24] gave a three-phase HDP model that proposed Mahalanobis distance-based class imbalance learning (CIL) framework for dealing with class imbalance problem (CIP) in the source dataset, as well as a classification and regression trees (CART)-based ensemble learning methodology for finding the best subset of the source dataset for metric matching. To check the feasibility of the proposed approach, the authors used nine projects from three public domain software defect repositories and compared them to four known advanced approaches. The findings of the experiments show that the proposed approach is more reliable in terms of AUC.

Wu et al. [40] presented the multi-source heterogeneous cross-project defect prediction (MHCPDP) approach in 2021, which employed AE to extract intermediate features from the original datasets rather than merely deleting redundant and unrelated features. To limit the impact of negative transfers and improve the performance of the classifier, the MHCPDP developed a multi-source transfer learning algorithm. The authors tested MHCPDP on five open source datasets in depth. The results of the experiments revealed that MHCPDP not only improved two performance measures but also addresses the drawbacks of traditional HCPDP approaches.

3 Proposed Defect Prediction Model

While HoCPDP allows more than one homogeneous project's datasets for training and testing of DP model, on the contrary, HCPDP-AE begins its DP process with a pair of datasets as source dataset S_{a*b} and target dataset T_{p*q}. Each row and column represents an instance and a software metric, respectively in both datasets.

Preprocessing of datasets is performed in the very first phase to make them compatible for their employment in the model as shown in Fig. 3. This phase entails the treatment of missing values, the labeling of a categorical/dependent variable, eradication of the CIP in an imbalanced training dataset, and the normalization of a given collection of data values. The far disproportionate ratio of instances in two groups referred to as CIP is the most challenging and significant problem that should be addressed in this phase. If one wants to use data resampling techniques [25] to tackle CIP, so there would be two ways to equalize the number of cases in majority and minority class. In the first way termed as random over-sampling (ROS), one attempts to generate more synthetic minority class observations, and in the other way termed as random under-sampling (RUS), one tries to minimize majority class observations in order to achieve an equivalent count of observations in both classes for a binary classification problem [26].

But, as per the state of the art, both approaches have drawbacks of their own [26]. The former approach induces redundant observations for minority class that can overgeneralize the minority class without taking into account the distribution of instances in the majority class. On the other hand, the latter method can exclude some helpful or relevant observations from the majority class without taking into consideration their significance in predicting the expected outcome.

Therefore, a novel hybrid approach named as chunk balancing algorithm (CBA) is proposed in the research study to treat CIP in order to obtain benefits as well as to overcome the shortcomings of both ROS and RUS techniques. SMOTE's overgeneralization and over-fitting issues are addressed in CBA by balancing minority class instances via creating chunks as stated in CBA, rather than introducing synthetic examples that cause duplication in the dataset distribution. RUS, on the other hand, eliminates majority class occurrences regardless of their significance in predicting the final outcome. To create n balanced chunks, CBA also entails the random selection of instances from the majority class. However, it provides all majority class instances an equal chance to become a part of the model's training rather than discarding them entirely. In this manner, CBA overcomes the constraints of data re-sampling methodologies used to handle CIP. The first phase of the HCPDP-AE model is completed throughout this way.

This algorithm takes an imbalanced dataset as an input and returns n chunks with almost equal numbers of instances from both the majority and minority groups as the output.

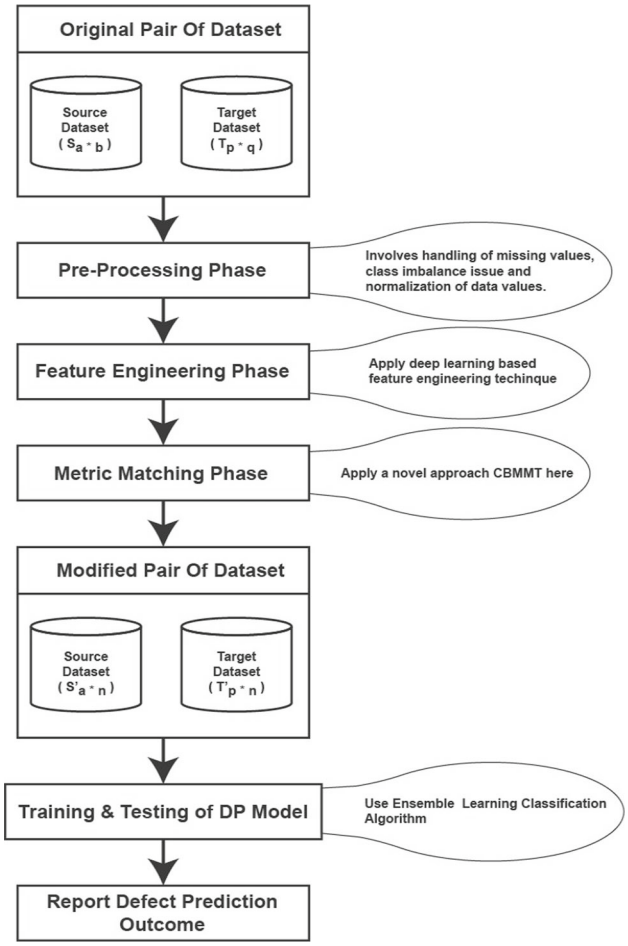


Fig. 3 Four-phased HCPDP-AE framework

Algorithm 1. Chunk Balancing Algorithm (CBA)

Input: Imbalanced Dataset D_{p*q} with p & q number of imbalanced observations and software features respectively.

Output: n balanced chunks & each chunk contains nearly equal count of observations from both categories.

1. Randomly shuffle all observations in dataset D;
2. Split up D into two subsets S_1 & S_2. Each subset is depicting the majority & minority category respectively;
3. $n_1 \leftarrow$ count of observations of majority category;
4. $n_2 \leftarrow$ count of observations of minority category;
5. $n \leftarrow \left\lceil \frac{n1}{n2} \right\rceil$;
6. Split up S_1 into n chunks of nearly same size;
7. **for** each chunk C_i such that i = 1 to n do,
8. | Merge S_2 with C_i in order to form a balanced chunk C'_i of size nearly equals to $2n_2$;
9. **end for**
10. return n balanced chunks (C'_1, C'_2, C'_3,……,C'_n);

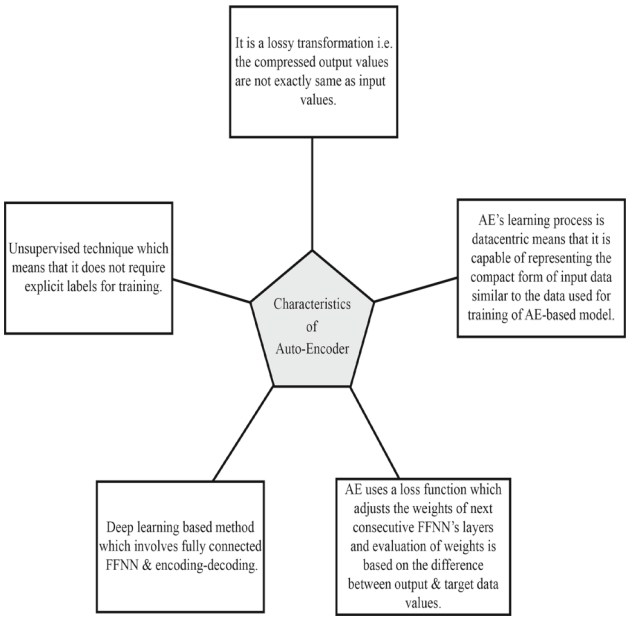


Fig. 4 Characteristics of auto-encoder

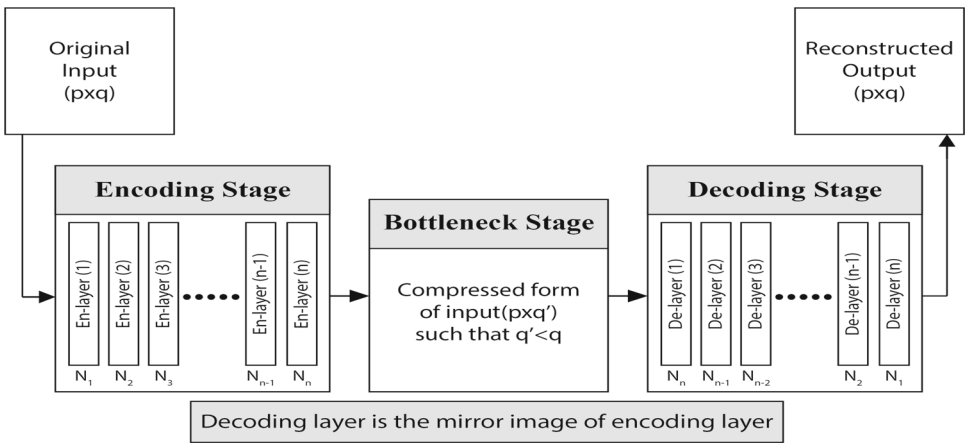
Today's age is known as era of data. So, it is prime need of time to extract meaningful, relevant and highly discriminating data that has higher significance in comparison with excluded data from pool for prediction of the expected results. The second phase of the model, i.e., feature engineering (FE), focuses on the same problem. It involves both feature selection and feature extraction [26]. Feature selection methods help to delete unnecessary and outdated features that are not crucial in determining expected outcomes. However, by introducing new lower-dimensional feature set along with discarding the original higher-dimensional feature set, feature extraction helps to decrease the dimensionality of the given pool of features. The accuracy and reliability of a SDP model is therefore strongly influenced by the collection of the most suitable and substantial features in the FE process. Deep learning-based FE techniques are not yet well explored with robust metric matching technique in the prediction of defects among projects with homogeneous and heterogeneous features, as per the literature survey carried out for the research analysis. So, auto-encoder (AE), a deep learning-based FE technique is applied to implement the second phase of HCPDP-AE model. The principle behind the use of the AE technique is to extract a feature set of lower dimension that can reproduce the original input using the encoding–decoding model. The purpose of using this technique is an unsupervised learning technique; it also employs feed-forward neural network (FFNN) for compact representation of original input known as representation learning [27]. In contrast to other related approaches, it provides better outcomes if some of the features used in the dataset have some kind of correlation between them rather than being entirely independent. The more detail of AE method is shown in Fig. 4.

All of the properties listed in Fig. 4 are implemented in the proposed HCPDP_AE model. For example, using the hidden layer's features, the output features cannot be determined exactly the same as the input feature set. As demonstrated in Table 6, the model generates varied RMSE values depending on the reduced number of features at the hidden layer for a given source dataset. This shows that the model's transformation of a higher-dimensional feature set to a lower-dimensional feature set is always ***lossy***.

The weights of each link between a hidden layer and an output layer or an input layer and a hidden layer at the next iteration are calculated by the weights of links between corresponding layers at the previous iteration and with the ***loss function*** (RMSE) that is used to estimate the transformation loss.

The model only accepts vectors of feature values as input, with no class labels, means HCPDP_AE is adhering to the *unsupervised nature* of the AE approach. It simply tries to learn a function that maps the higher-dimensional input x to the lower-dimensional input y, and then attempts to recreate x using y.

Fig. 5 Three stage architecture of auto-encoder



The model uses a *deep learning strategy* to extract the features, but only one hidden layer was employed because one hidden layer is adequate to train the FFNN considering the feature cardinality of employed datasets.

The detailed encoding–decoding architecture of AE is well shown in Fig. 5. It consists of three constituent stages: encoding stage, bottleneck stage and decoding stage. In the encoding stage, it is possible to have n numbers of encoding layers that are En-LAYER(1), En-LAYER(2), En-LAYER(3),…, En-LAYER(n) consisting of N_1, N_2, N_3,…, N_n number of nodes in each respective layer. This stage encodes the original input with q number of features in a compact form with q' number of features such that q' < q.

The best possible compressed form of input features that are used to train the DP model is given by the second stage, i.e., bottleneck stage. In the last stage, the architecture attempts to recreate the original input from the compressed form generated from the bottleneck stage and aims to regularize the loss of reconstruction by comparing original data and reconstructed data. The decoding step can be thought of as a mirror image of the encoding stage, with De_LAYER(1) performing the mirror operations as done in En_LAYER(n) with N_n nodes. During back propagation in architecture, the model's training focuses to mitigate the reconstruction loss. In this way, the proposed research aims to incorporate the second phase of the model by investigating the same using FE methodology based on deep learning.

The encoding and decoding equations for the AE model with one hidden layer are described in Eqs. (1) and (2), respectively, where F_A is the encoding function that maps higher-dimensional input A to compressed form B and $G_{\mathbf{B}}$ is the decoding function that attempts to recreate input as A' from compressed form B with minimal reconstruction loss. A_f is the applied activation function and w and w' are the weighting parameter in encoding and decoding stage, respectively. The bias values for both stages are denoted by b_A and $b_{B,}$ respectively.

$$B = F_{\lambda}(A) = A_f(wA + b_A) \tag{1}$$

$$A' = G_{\lambda'}(B) = A_f\left(w'B + b_B\right) \tag{2}$$

The main aim of AE is to reduce reconstruction loss on an original input A, which is well defined by the objective function λ as follows:-

$$\lambda = \min L_f\left(A, A'\right) \tag{3}$$

where $A' = G(F(A))$ and L_f is the loss function depending upon the type of reconstruction (linear or nonlinear). It produces the optimal set of weights for mapping the input variables to the target or output variable with the least amount of reconstruction loss. The loss function used for neural network's implementation is strongly intertwined to the selected activation function. According to [28], the best result for a neural network designed for a regression problem is given by rectified linear activation function (ReLAF) with root mean squared error (RMSE) loss function. Table 1 offers more information on the loss function and activation function used in executing the AE Model for the comprehensive research.

Since the built model's training accuracy is primarily based on the matched metric collection, the third stage of heterogeneous prediction modeling, i.e., metric matching, is the most critical and complex stage. To illustrate the association between metrics, modern methods such as the least square method, dispersion diagram method and Spearman's rank correlation method can be used. After measuring the coefficient of correlation value (CCV) between different possible feature pairs of two applications, the model selects those feature pairs whose CCV is greater than the given cutoff threshold.

After imposing a cutoff threshold filter, a set of feature pairs known as strongly correlated features is selected. The source dataset S cannot be used to model defects in a heterogeneous target dataset T if the highly correlated metric collection for a pair of datasets (S, T) is null. As per the literature study [15, 18], the authors picked instances (rows in a dataset) at random and created only one training chunk from the instances pool of a dataset with more number of instances in contrast to another dataset, which might be a source or target dataset. However, it is likely that arbitrarily picking instances from either of the datasets.

at once would result in poor metric matching between feature pairs, or that the count of strongly correlated feature pairs obtained after applying the threshold filter would be insufficient to train the DP model.

To address this shortcoming of traditional metric matching approach, a novel metric matching method known as chunk-based metric matching technique (CBMMT) is proposed in this article to incorporate this most critical and hot part of the heterogeneous prediction system. CBMMT being also random in nature checks each chunk of instances for metric matching and selects the chunk for DP model's training that shows the maximum number of strongly correlated feature pairs with CCV greater than the threshold. In this manner, the most important step of the HCPDP-AE is effectively implemented with CBMMT.

Table 1 Loss function and activation function

Type of function	Name of function	Description	Mathematical formula
Activation function	ReLAF	Convolutional neural networks and deep learning often use ReLAF for regression problem. It is rectified from the bottom. This function has a range of zero to infinity	$A_f(z) = \max\{0, z\}$ (4) where z is an input variable to activation function
Loss function	RMSE	The prediction performance of a regression model is measured by RMSE. The root taken over the sum of squared distances between our target variable and predicted values is defined as RMSE	$L_f = \sqrt{\frac{\sum_{i=1}^{n}(yi-xi)^2}{n}}$ (5) where x_i and y_i are true and predicted values, respectively, and n is total number of inputs

Algorithm 2. Chunk-Based Metric Matching Technique (CBMMT)

Input: Source Dataset S_{a*b} & Target Dataset T_{p*q} such that both datasets have equal number of software metrics (i.e., b=q & assume, b=q=n) and count of training observations should be equal or greater than the observations count of validation dataset (i.e., $a \geq p$).

Output: (1) Correlation Matrix C^k_{n*n}, where each C^k_{i*j} entry denotes the association value between i^{th} feature in k^{th} chunk of source dataset & j^{th} feature in target dataset such that $-1 \leq C_{i*j} \leq 1$.
(2) k^{th} chunk of source dataset as training dataset.
(3) Set of strongly correlated features between source & target dataset with respective CCV.

1. $S_{a*b} \leftarrow$ Source Dataset;
2. $T_{p*q} \leftarrow$ Target Dataset;
3. $a \leftarrow$ count of observations in source dataset;
4. $p \leftarrow$ count of observations in target dataset;
5. Randomly shuffle all instances of source dataset;
6. $m \leftarrow \left\lfloor \frac{a}{p} \right\rfloor$ be the number of candidate chunks of source dataset that can be used for metric matching;
7. **for** k = 1 to m
8. Calculate Correlation Matrix C^k_{n*n} using Spearman's Rank Correlation Technique (SRCT) for each chunk of source dataset with target dataset;
9. **end for**
10. return k^{th} Correlation Matrix showing higher number of association values $C_{i*j} \geq$ threshold;
11. Use CBA to handle CIP in k^{th} source dataset's chunk & return n balanced chunks as training datasets to train the DP model;
12. return correlated feature set with CCV;

The model is trained using an appropriate machine learning algorithm after identifying this strongly correlated metric collection, and the performance results are reported in the model's final step. Different evaluation metrics are used to outline the output findings. The traditional DP approach, known as WPDP, uses only one dataset, which is then split up into two components, one for training and another for testing, as per the partition strategy (7:3 or 6:4). Fig. 6 presents the structured WPDP model in detail.

The function of all three phases is identical to that of the HCPDP-AE model. In this study, the performance of both categories of DP (WPDP & HCPDP-AE) is evaluated using both single as well as ensemble learning approaches. When opposed to a single model, the ensemble learning approach allows for the improved predictive results.

4 Datasets Used and Performance Measures

4.1 Data Collection

The study uses 16 publicly available and widely used datasets from four open source repositories, including AEEEM, ReLink, SOFTLAB and NASA, for the experimental analysis. The dataset AEEEM was created by D'Ambros et al. [38]. There are 61 metrics in all, including 17 source code metrics, 5 past-defect metrics, 5 entropy-of-change metrics, 17 entropy of-source-code metrics, and 17 churn-of-source-code metrics [38]. AEEEM contains linearly decaying entropy (LDHH) and weighted churn in particular (WCHU).

The Understand tool yielded 26 findings of coding consequences, which were stored in the ReLink repository. Wu et al. [15] gathered manually validated and rectified defect data in ReLink. The number of instances in its three datasets varied from 56 to 399, but the number of features is fixed at 26 [15].

The NASA data was obtained over a five-year period from a variety of NASA contractors working in various geographic locations across the USA [5]. Size, readability, complexity, and other static code metrics for NASA datasets are all strongly connected to software quality.

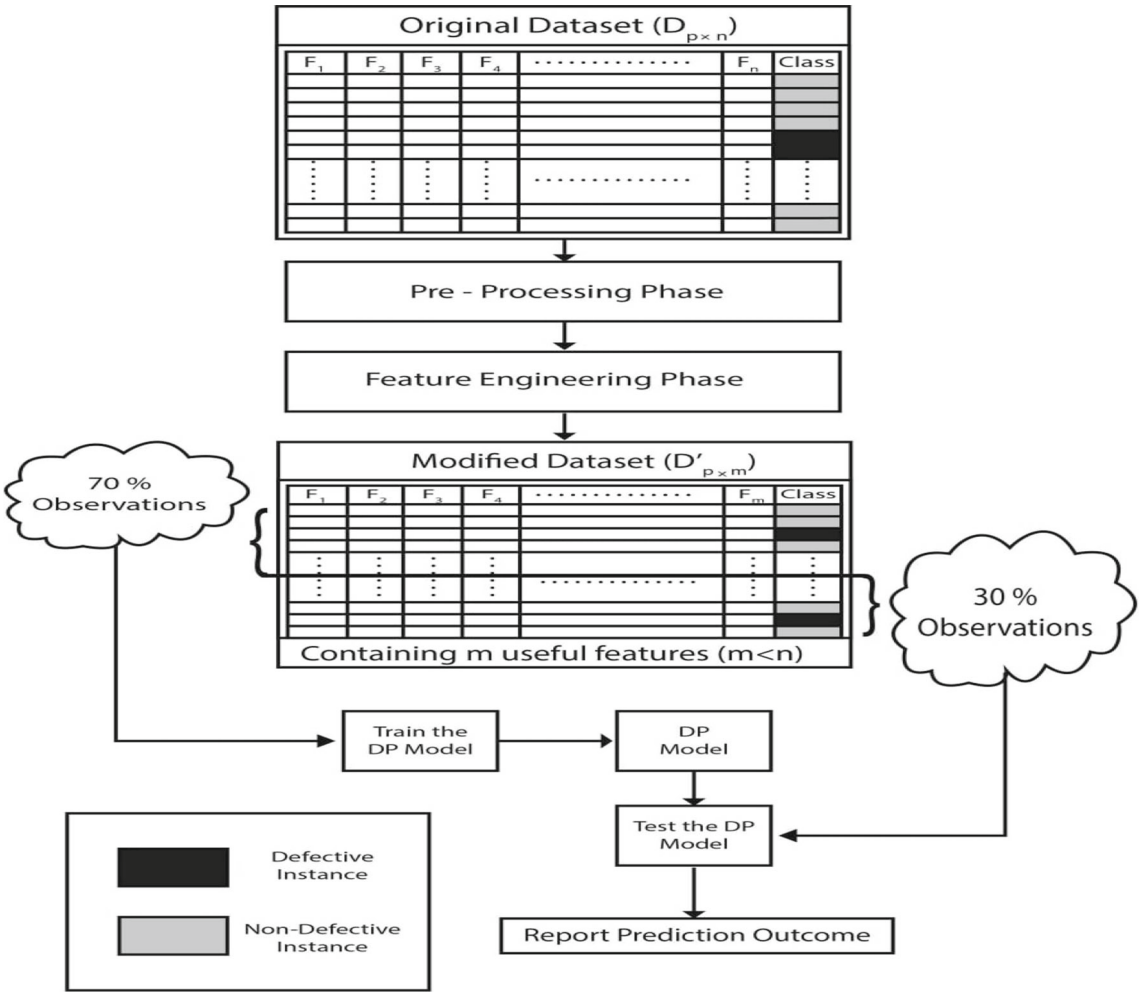


Fig. 6 Three-phase WPDP framework

The five datasets for SOFTLAB repositories were obtained by a Turkish software industry (SOFTLAB). Each dataset contains data about controller software for a variety of electrical appliances. It also has patented datasets that consists of Halstead and McCabe cyclomatic metrics. The used SOFTLAB and NASA datasets were obtained from the PROMISE repository [39]. There are 28 metrics that these two projects have in common.

All experimental studies of SDP are carried out on these repositories based on a thorough literature review [14, 15]. Second, the datasets must contain a sufficient number of features in order to execute effective deep learning-based feature extraction and metric matching. Some datasets, such as MORPH, have only 20 features, which are insufficient for the proposed research study to be implemented. As a result, these repositories were picked by the authors for their research study. There is no other particular motive for choosing these datasets.

Table 2 provides additional information about these datasets. According to Table 2, the proportion of defective instances in four project categories ranges from 7.43% to 50.51%. The class imbalance ratio (CIR) is the ratio of the number of defective instances to the number of non-defective instances, or vice versa. In a particular training sample, the lower will be the CIR value; the greater will be the imbalance problem (IP). CIR values range from 6.79 (highest IP) to 102.08 (lowest IP) in datasets CM1 and Apache, respectively. On the other hand, there are 61, 26, 29 and 37 software metrics in four repositories, respectively. The source of all datasets is given as:

https://github.com/Sanuj12/ROHIT_VASHISHT.git

4.2 Performance Parameters

In this section, the various metrics used to gauge the efficiency of different machine learning classifiers are stated. During the evaluation process, the other parameters in the confusion matrix are used. Table 3 indicates the confusion matrix that was used to estimate erroneous classifications.

Table 2 Datasets illustration

Project group	Datasets	Count of instances			Class imbalance ratio (CIR)	Count of software metrics
		Total	Defective	Non-defective		
AEEEM [38]	EQ	324	129 (39.81%)	195 (60.19%)	66.15	61
	JDT	997	206 (20.66%)	791 (79.34%)	26.04	
	LC	691	64 (9.26%)	627 (90.74%)	10.20	
	ML	1862	245 (13.15%)	1617 (86.85%)	15.15	
	PDE	1492	209 (14.01%)	1283 (85.99%)	16.29	
ReLink [15]	Apache	194	98 (50.51%)	96 (49.49%)	102.08	26
	Safe	56	22 39.28%)	34 (60.72%)	64.71	
	Zxing	399	118 (29.57%)	281 (70.43%)	41.99	
SOFTLAB [39]	ar1	121	9 (7.43%)	112 (92.57%)	7.43	29
	ar3	63	8 (12.69%)	55 (87.31%)	14.55	
	ar4	107	20 (18.69%)	87 (81.31%)	22.99	
	ar5	36	8 (22.22%)	28 (77.78%)	28.57	
	ar6	101	15 (14.85%)	86 (85.15%)	17.44	
NASA [5]	CM1	327	42 (12.84%)	285 (87.16%)	6.79	37
	MW1	253	27 (10.67%)	226 (89.33%)	8.37	
	PC1	705	61 (8.65%)	644 (91.35%)	10.56	

Table 3 Confusion matrix

Actual outcome	Predicted outcome	
	Defective	Non-defective
Defective	True positive (TP)	False negative (FN)
Non-defective	False positive (FP)	True negative (TN)

- Accuracy:—The ratio of true outcomes (TP and TN) to the total number of instances examined is known as accuracy. Its value varies from 0 to 1, with 0 representing the least accurate result and 1 representing the most accurate result.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6}$$

- Recall (Rate of True Positives):—It is also known as sensitivity and defined as the chance of getting a positive test if an instance is defective or non-defective.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

- F-Score:-It is also known as the F-measure, because it is a measurement of a test's accuracy in a statistical study of binary classification. In order to measure the score, it considers both the test's accuracy and precision. The harmonic mean of precision (p) and recall (r) are used to determine it as per Eq. (8).

$$\text{F-Score} = \frac{2*\text{p}*\text{r}}{\text{p} + \text{r}} \tag{8}$$

Table 4 Relation between AUC value and prediction performance

AUC value	Prediction performance
0	The model will predict positive case as negative and vice versa. (Worst Case)
0.5	The model is having trouble in distinguishing between positive and negative scenarios
Between 0.5 and 1	Positive cases have a higher chance of being classified correctly by the model than negative cases
1	The model will predict positive case as positive and vice versa. (Perfect case)

- Area Under Operating Curve (AUC):- It is a plot of true positive rate (TPR) and false positive rate (FPR) that is used to determine the overall efficacy of a classification algorithm. The classification model would be more accurate if the AUC parameter is set to a higher value. For a given classification algorithm, the maximal AUC value is 1. In Table 4, the prediction performance corresponding to the AUC value is discussed in greater detail.

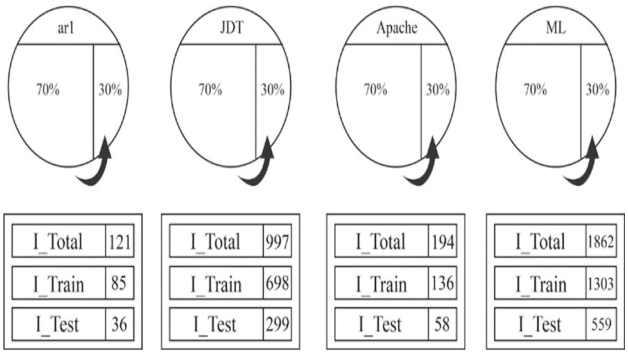


Fig. 7 With-In project defect prediction

Table 5 Prediction pairs for WPDP

Prediction pair	Training dataset	Testing dataset
WPDP-P1	ar1	ar1
WPDP-P2	JDT	JDT
WPDP-P3	Apache	Apache
WPDP-P4	ML	ML
WPDP-P5	ar3	ar3
WPDP-P6	EQ	EQ
WPDP-P7	ar4	ar4
WPDP-P8	PC1	PC1
WPDP-P9	MW1	MW1
WPDP-P10	CM1	CM1

5 Experimentation Setup

The main goals of the proposed research study are divided into four groups. The study's first two goals compare the performance of the respective HCPDP's and WPDP's frameworks with and without the use of the FE technique. The third objective is to see if and how well the prediction performance of HCPDP-AE model is comparable to the prediction efficiency of WPDP for a given set of machine learning classifiers. Lastly, the study's final aim is to compare the performance of the proposed model to that of the existing benchmarked models, as well as to validate the former. The research analysis performed two experiments in order to address the stated four research questions.

5.1 Experiment 1

The aim of this experiment is to look into the conventional category of DP, namely WPDP's output with two categories of FE techniques that are data-based FE and deep learning-based FE. To begin, the dataset is preprocessed to remove unnecessary software features and encode the categorical data with the tag. The process assigns 0 or 1 to the defective and non-defective cases, respectively. As well as, it also employs the novel CBA mentioned in Sect. 3 to deal with CIP in imbalanced training datasets. According to the current state of the art [23, 29], DP on heterogeneous projects has yet to be examined using FE based on deep learning in conjunction with effective and resilient strategies for dealing with CIP and for executing metric matching, which is the fundamental phase of the HCPDP framework. Data-driven FE is a generic methodology that does not require domain awareness. Such a representation is found by mining pair-wise feature correlations, evaluating the linear or nonlinear relationship between each pair, applying regression, and choosing the most stable relationships [30]. The aim of the experiment is to compare the performance of a three-stage WPDP model using two FE approaches: regression feature engineering (RFE) and AE, respectively [27, 30]. RFE and AE techniques are used as data-based and deep learning-based FE methods, respectively, to implement the second phase of the model. After the selection of the most discriminating features, the available number of instances is segregated into training and validation instances set in a ratio of 7:3. Figure 7 depicts software defect predictions within a project where I_Total, I_Train and I_Test denote count of total, training and testing observations in a dataset, respectively. Table 5 lists the training and testing datasets that have been used to conduct the experiment.

Table 6 Prediction combinations for HCPDP

Prediction pair	Source dataset	Target dataset
HCPDP-P1	JDT	ar1
HCPDP-P2	ar5	JDT
HCPDP-P3	ar1	Apache
HCPDP-P4	Safe	ML
HCPDP-P5	EQ	ar3
HCPDP-P6	LC	ar3
HCPDP-P7	CM1	EQ
HCPDP-P8	PC1	ar4
HCPDP-P9	LC	PC1
HCPDP-P10	EQ	MW1

5.2 Experiment 2

The aim of the experiment is to see how the FE step affects the performance of the proposed four-phase HCPDP-AE framework when using the stacking-based ensemble classification method. To carry out this experiment, ten prediction pairs mentioned in Table 6 are taken from four open source projects: AEEEM, ReLink, NASA and SOFTLAB. Prediction pairs are formed based on the number of maximal-associated feature pairs found between them.

Preprocessing of datasets involves the deletion of redundant data and the encoding of categorical data by labels in the first phase. In this phase, class imbalance learning (CIL) is also applied to deal with the significant difference in the ratio of binary type instances count. To carry out CIL, the model employs a novel hybrid approach known as CBA as mentioned in Sect. 3. Following that, feature ranking and feature selection strategies are used to derive a list of K- best features that are more relevant in the prediction of desired final outcome for a given dataset of features. The research aims to examine the framework using AE as a deep learning-based FE approach. The features are chosen to make the two datasets dimensionally identical, allowing for easy metric matching. After the selection of useful features, the metric matching method evaluates the relationship between each combination of feature pair of source and target dataset. CBMMT is used to execute the third phase, which provides a set of highly correlated features for a given heterogeneous prediction pair.

To train the HCPDP-AE model, i.e., to execute the final modeling phase, the experiment incorporates ensemble classification approach. Finally, the efficiency of the model is assessed using performance parameters specified in Sect. 4.

6 Results and Discussion

TensorFlow 2.0 with GPU support is used to conduct both experiments, which run on a Windows 10 operating system with an Intel Core i5-1130G7 processor and 32GB RAM. The results of the experiments are discussed in this section. Tables 7, 8, 9, 10, 11, 12, 13, 14 and 15 and Figs. 8, 9, 10, 11, 12 and 13 display the experimental results. The analysis uses precision, recall, F-score (FS), and AUC as performance indicators. Randomness, class imbalance issue, and prediction threshold, all have a significant impact on the model's accuracy and recall. As a result, tenfold cross-validation (CV) is utilized to measure these factors in order to obtain results that are less skewed and have a low variance. Other parameters like AUC and F-score are also evaluated on basis of 30 repeated trials to reduce effect of randomness on prediction performance. Second, these parameters (AUC and FS) have no impact due to imbalanced distribution of instances and prediction threshold.

6.1 RQ1. Compare and contrast the use of data-driven and deep learning-based FE strategies with the conventional approach to DP, i.e. WPDP.

To compare the WPDP's output using RFE and AE, as data-driven and deep learning techniques, respectively. Ten with-in prediction pairs WPDP-P1 to WPDP-P10 have been

Table 7 Feature extraction using AE

Dataset	Number of Neurons		RMSE ($*10^{-3}$)
	Input layer	Hidden layer	
ar1	29	20	188.2
		19	234.1
		18	231.4
		17	**137.1**
		16	178.3
		15	280.9
JDT	61	40	201.8
		35	216.3
		30	188.8
		25	123.5
		20	**120.7**
		15	128.6
Apache	26	20	188.7
		19	140.9
		18	133.6
		17	160.1
		16	**126.1**
		15	192.4
ML	61	40	178.3
		35	167.7
		30	165.2
		25	130.2
		20	124.6
		15	**112.9**

Table 8 Feature reduction for WPDP

Dataset	Count of features		Percentage reduction
	Before FE	After FE	
ar1	29	17	41.37
JDT	61	20	67.21
Apache	26	16	38.46
ML	61	15	75.41
ar3	29	17	41.37
EQ	61	24	60.66
ar4	29	16	44.83
PC1	37	22	40.54
MW1	37	19	48.65
CM1	37	20	45.95
Average reduction (%)			**50.45**

Table 9 Comparison of WPDP performance

Prediction Pair	CIL	Evaluation baseline											
		Without FE				With RFE				With AE			
		Acc	Recall	AUC	FS	Acc	Recall	AUC	FS	Acc	Recall	AUC	FS
WPDP-P1	SMOTE	0.60	0.67	0.429	0.53	0.64	0.70	0.421	0.61	0.70	0.83	0.598	0.68
	RUS	0.59	0.60	0.396	0.51	0.71	0.75	0.467	0.55	0.81	0.86	0.642	0.60
	CBA	0.71	0.78	0.567	0.59	0.74	0.83	0.602	0.65	0.89	0.94	0.739	0.74
WPDP-P2	SMOTE	0.62	0.60	0.516	0.59	0.76	0.71	0.555	0.63	0.87	0.83	0.631	0.68
	RUS	0.69	0.67	0.592	0.62	0.79	0.75	0.646	0.69	0.84	0.81	0.600	0.72
	CBA	0.73	0.71	0.606	0.68	0.80	0.87	0.671	0.74	0.90	0.89	0.702	0.81
WPDP-P3	SMOTE	0.65	0.58	0.501	0.54	0.69	0.64	0.539	0.63	0.83	0.79	0.611	0.70
	RUS	0.62	0.51	0.431	0.49	0.71	0.70	0.570	0.57	0.88	0.83	0.667	0.66
	CBA	0.69	0.68	0.587	0.61	0.82	0.80	0.618	0.73	0.94	0.89	0.758	0.86
WPDP-P4	SMOTE	0.59	0.68	0.512	0.59	0.69	0.71	0.544	0.62	0.79	0.85	0.683	0.68
	RUS	0.63	0.73	0.624	0.60	0.70	0.76	0.640	0.69	0.75	0.80	0.622	0.76
	CBA	0.69	0.78	0.659	0.68	0.80	0.88	0.712	0.79	0.87	0.96	0.810	0.88
WPDP-P5	SMOTE	0.55	0.56	0.388	0.58	0.61	0.55	0.400	0.70	0.72	0.68	0.567	0.78
	RUS	0.41	0.51	0.316	0.44	0.50	0.60	0.376	0.59	0.68	0.71	0.601	0.76
	CBA	0.70	0.68	0.566	0.69	0.79	0.65	0.701	0.78	0.85	0.88	0.834	0.89
WPDP-P6	SMOTE	0.43	0.55	0.404	0.59	0.55	0.59	0.501	0.64	0.70	0.68	0.613	0.75
	RUS	0.51	0.51	0.399	0.54	0.60	0.57	0.493	0.65	0.78	0.72	0.600	0.87
	CBA	0.69	0.60	0.545	0.67	0.76	0.63	0.616	0.72	0.90	0.92	0.789	0.94
WPDP-P7	SMOTE	0.33	0.45	0.300	0.50	0.57	0.55	0.378	0.59	0.65	0.61	0.489	0.67
	RUS	0.37	0.50	0.317	0.48	0.50	0.58	0.399	0.60	0.62	0.69	0.606	0.68
	CBA	0.55	0.59	0.487	0.55	0.61	0.65	0.525	0.70	0.78	0.74	0.778	0.84
WPDP-P8	SMOTE	0.37	0.49	0.445	0.43	0.52	0.55	0.569	0.59	0.62	0.70	0.615	0.71
	RUS	0.35	0.55	0.487	0.51	0.55	0.50	0.606	0.64	0.69	0.74	0.688	0.76
	CBA	0.48	0.62	0.551	0.65	0.67	0.71	0.690	0.81	0.83	0.84	0.792	0.88
WPDP-P9	SMOTE	0.45	0.51	0.443	0.55	0.55	0.59	0.501	0.62	0.64	0.77	0.611	0.73
	RUS	0.40	0.55	0.488	0.49	0.62	0.68	0.529	0.74	0.72	0.77	0.589	0.85
	CBA	0.51	0.58	0.526	0.61	0.73	0.80	0.633	0.82	0.88	0.85	0.712	0.88
WPDP-P10	SMOTE	0.40	0.38	0.311	0.45	0.55	0.52	0.467	0.65	0.64	0.66	0.688	0.78
	RUS	0.37	0.35	0.347	0.50	0.59	0.61	0.544	0.72	0.62	0.66	0.701	0.77
	CBA	0.51	0.59	0.498	0.65	0.62	0.75	0.659	0.83	0.89	0.82	0.870	0.90

used for the analysis. In the first step of WPDP model, i.e., preprocessing, binary encoding is used to label the target variable and Z-score normalization (ZSN) is used to scale all feature values so that one particular feature does not overshadow the others. The mean and standard deviation of each feature were determined to produce normalized values. In this manner, feature encoding and feature scaling were performed as two subtasks under this step. The next and last subtask in preprocessing step is to manage CIP in all four prediction pairs' training datasets. The range of CIR values for all ten with-in prediction pairs is shown in Table 2. With a CIR of 6.29%, CM1 is the most imbalanced dataset, whereas Apache has the most balanced data distribution with a CIR of 102.08%. Traditional data re-sampling methods used for CIL have their own pitfalls, as discussed in Sect. 3.

In the analysis, the prediction accuracy of the WPDP model is assessed using three CIL approaches: Synthetic Minority Over-Sampling Technique (SMOTE) as a ROS technique, RUS and a novel approach CBA [37]. The algorithms SMOTE_CIL and RUS_CIL, respectively, explain the detailed procedure for implementing SMOTE and RUS.

Table 10 Statistics of feature reduction for HCPDP

Dataset	Count of features		Feature reduction (%)
	Before FE	After FE	
JDT	61	20	67.21
ar5	29	22	24.14
ar1	29	17	41.37
Safe	26	20	23.08
EQ	61	22	63.93
LC	61	18	70.49
Apache	26	16	38.46
ML	61	15	75.41
ar3	29	20	31.03
ar4	29	16	44.83
MW1	37	19	48.65
CM1	37	20	45.95
PC1	37	22	40.54
Average reduction (%)			**47.31**

Algorithm 3. SMOTE_CIL

Input: - Imbalanced Dataset D

Output: - Balanced Dataset D' with equal number of instances in both majority and minority class.

1. Identify the minority and majority class in D.
2. Evaluate the OS percentage (OS %) by using equation (9) where IC_{Maj} and IC_{Min} are the count of the majority and minority class respectively.

$$OS\% = \left\lceil \frac{ICmaj - ICmin}{ICmaj} * 100 \right\rceil \quad (9)$$

3. As per equation (10), estimate the Count of Replica Required (CRR) to be created for the minority class for balancing CIR. (such that $IC_{maj} \approx IC_{Min} + CRR$)

$$CRR = \left\lceil \frac{OS\% * ICmaj}{100} \right\rceil \quad (10)$$

4. Select randomly any one minority class instance and find its closest neighbour.
5. Estimate the disparity between the random instance picked in step 4 and one of its closest neighbours.
6. Calculate additive disparity by multiplying estimated disparity in step 5 by a random number generated using a random number generator in the range 0 to 1.
7. Create a new minority class instance's replica by adding the value of a randomly chosen random instance in step 4 with additive disparity.
8. Repeat steps 4 to 7 until the required number of minority class replicas are generated based on the OS percent.

Algorithm 4. RUS_CIL

Input: - Imbalanced Dataset D
Output: - Balanced Dataset D' with equal number of instances in both majority and minority class

1. Identify the minority & majority class in D.
2. Evaluate the US percentage (US %) by using equation (11) where IC_{Maj} & IC_{Min} are the count of the majority and minority class respectively.

$$US\% = \left\lceil \frac{ICmaj - ICmin}{ICmaj} * 100 \right\rceil \quad (11)$$

3. Estimate the Count of Instances to be Eliminated (CIE) from the majority class to balance CIR as per the equation (11). (such that $IC_{Min} \approx IC_{Maj}$ - CIE)

$$CIE = \left\lceil \frac{US\% * ICmaj}{100} \right\rceil \quad (12)$$

4. According to the count estimated in step 3, remove a majority class instance one by one at random.

For example, in HCPDP_P3, the source dataset ar1 has CIR as 7.43. On the basis of count of instances as seen in Table 2, the minority and majority classes are identified as defective and non-defective category, respectively. The OS% can be computed as 92% using the values IC_{Maj} as 112 and IC_{Min} as 9, according to Eq. (9). According to Eq. (10), the CRR is 103, i.e., in order to attain CIR as 1:1, 103 additional replicas should be created for defective class by following steps 4 to 7 in SMOTE_CIL. To deal with CIP, RUS_CIL attempts to delete any 243 random instances from the non-defective class in dataset CM1.

The shallow AE model is used to extract features using a three-layer convolutional FFNN. The count of original features in a dataset is equal to the number of neurons in both input and output layers. The hidden layer's number of neurons is determined by the rules-of-thumb that represents the compact number of features [24]. The reconstruction loss for respective output features that are reconstructed from collection of both extracted features and initial input feature set is represented by RMSE. The output of neurons in the hidden layer with the lowest RMSE would be considered as final set of extracted features. The extraction of features is performed in this way using a deep learning-based AE model. For example, Table 7 displays the RMSE values for the considered set of neurons in the hidden layer for first four with-in prediction pairs. It shows that the reduced number of features for datasets ar1, JDT, Apache and ML are 17, 20, 16 and 15 respectively, with corresponding least RMSE of 0.1371, 0.1207, 0.1261 and 0.1129.

The statistics of the original and extracted features using AE model is shown in Table 8. It indicates an overall decrease of 50.45% in the total number of input features.

As shown in Fig. 7, 70% and 30% of the total observations are used as training and testing observations for each prediction pair, respectively. For example, WPDP-P1 (ar1) uses 85 instances to train the WPDP model and 36 instances to validate the proposed WPDP model. The WPDP model is trained using support vector machine (SVM) with linear radial basis function kernel (RBFK). For linear as well as nonlinear classification, SVM-RBF is the most useful and effective algorithm. It is beneficial when the number of features in a dataset is substantially greater than the number of instances. When looking at the cardinality of used datasets, the number of observations in Apache, ar1, ar3, ar4, ar5, ar6 and Safe is substantially lower. Due to the limited number of features in the training dataset, SVM was chosen to have the least or no impact on prediction performance. Table 9 shows the values of performance parameters after validating the model with 30% testing observations for all prediction pairs under three baselines that are without FE, with FE using RFE and AE.

The results of Table 9 can be viewed from two angles. The first aspect is a comparison of DP's output between standard CIL approaches and the proposed hybrid CBA approach. In contrast to SMOTE and RUS, the results reveal that CBA outperforms for all prediction pairs.

As per the statistics of Table 9, CBA offers the best values of all performance parameters with accuracy as 0.94 for WPDP-P3 and recall, AUC and FS as 0.96, 0.810 and 0.88, respectively, for WPDP-P4, when AE model was employed for feature extraction.

When CIP is handled using SMOTE with the original pool of features, the prediction pair WPDP-P7 has the lowest accuracy and AUC of 0.33 and 0.300, respectively. For all three baselines, SMOTE and RUS have given comparable results. The first baseline, i.e., WPDP without FE, has the lowest output among the three. Using the average values of all performance measures for all ten prediction pairings under the CBA strategy to combat CIP, Fig. 8 depicts the performance comparison of WPDP under three baselines as:

$$WPDP\ without\ FE < WPDP\ with\ RFE < WPDP\ with\ AE.$$

As a result, the proposed three-phase WPDP model (WPDP-AE) outperforms when a hybrid solution is used to tackle CIP and feature extraction is performed using a deep learning-based model rather than a data-driven approach.

6.2 RQ2. Compare the Prediction Performance of Proposed HCPDP-AE Framework with and Without FE Phase.

Ten heterogeneous prediction pairs (HCPDP-P1 to HCPDP-P10) from three open source projects are considered to test

Table 11 Performance comparison of HCPDP output

Prediction combination	HCPDP without FE (HCPDP)				HCPDP with FE (HCPDP-AE)			
	Accuracy	Recall	FS	AUC	Accuracy	Recall	FS	AUC
HCPDP-P1	0.67	0.70	0.62	0.518	0.87	0.85	0.78	0.717
HCPDP-P2	0.72	0.68	0.65	0.589	0.85	0.80	0.78	0.697
HCPDP-P3	0.61	0.56	0.52	0.572	0.87	0.90	0.89	0.781
HCPDP-P4	0.69	0.65	0.59	0.584	0.81	0.94	0.83	0.791
HCPDP-P5	0.59	0.61	0.55	0.598	0.79	0.73	0.77	0.788
HCPDP-P6	0.66	0.72	0.62	0.641	0.86	0.88	0.84	0.812
HCPDP-P7	0.52	0.61	0.65	0.516	0.72	0.77	0.81	0.777
HCPDP-P8	0.55	0.65	0.65	0.569	0.79	0.77	0.80	0.813
HCPDP-P9	0.59	0.71	0.69	0.605	0.80	0.85	0.90	0.890
HCPDP-P10	0.71	0.78	0.81	0.703	0.91	0.95	0.94	0.901
Average	**0.63**	**0.67**	**0.64**	**0.587**	**0.83**	**0.84**	**0.83**	**0.796**

Table 12 Execution time (in seconds) for HCPDP and HCPDP_AE

Prediction combination	HCPDP without FE (HCPDP)		HCPDP with FE (HCPDP-AE)	
	Training time	Classification time	Training time	Classification time
HCPDP-P1	2.375	0.121	3.519	0.101
HCPDP-P2	1.054	0.412	2.134	0.217
HCPDP-P3	0.972	0.055	1.463	0.031
HCPDP-P4	0.883	0.112	2.320	0.010
HCPDP-P5	1.886	0.051	3.188	0.058
HCPDP-P6	1.719	0.886	3.032	0.519
HCPDP-P7	0.871	0.122	1.101	0.111
HCPDP-P8	0.662	0.211	1.921	0.197
HCPDP-P9	1.034	0.651	4.045	0.335
HCPDP-P10	0.781	0.217	2.011	0.200
Average	**1.224**	**0.284**	**2.473**	**0.178**

the prediction accuracy of the proposed HCPDP-AE model with and without FE. Binary encoding and ZSN techniques are used for feature encoding and feature scaling, respectively, in the first step. The preprocessing step of the dataset is carried out in the same way as done in the WPDP scenario. The implementation of HCPDP-AE model is explained in detail using the prediction combination HCPDP-P1, where JDT and ar1 are the source and target dataset, respectively. In both datasets, the target variable is initially encoded by labeling defective instances as 0 and non-defective instances as 1. Feature extraction is now performed using the three-layer AE model in the same way as done in WPDP-AE model. Table 10 shows the total number of extracted features and feature reduction%age for all training datasets in ten prediction combinations that were considered for the analysis.

The extracted feature set is chosen based on least reconstruction loss in terms of RMSE value for a particular input feature set. The number of neurons in the hidden layers that produce the least RMSE is referred as the compressed number of features for that particular dataset.

After executing second phase of FE, JDT and ar1 now have dimensions of (997 $\times$ 20) and (121 $\times$ 17), respectively. In heterogeneous prediction, the next phase, metric matching, is the most important among all four phases. The HCPDP-AE model used a novel technique called CBMMT to accomplish this phase. The primary criterion of CBMMT is that the two datasets have the same cardinality of software features. The authors applied the Fisher Score (FS) method, a supervised feature ranking and selection technique to choose the best 17 features (minimum between 20 and 17) that are more relevant and useful in predicting the final outcome. There is no strong reason to use FS technique for choosing the optimal subset of features. FS evaluates each feature's significance independently and ranks them according to their utility in predicting the expected outcome. It generates a list of features sorted by their ranking values in descending order. After

that, CBMMT is used to do metric matching once the number of features has been equalized in each dataset. The execution of metric matching in HCPDP-P1 is shown in Fig. 9. According to CBMMT, the variables a, p, n and m have values of 997, 121, 17 and 8, respectively. That means there are total eight candidate chunks (JDT_C1, JDT_C2,……., JDT_C8) of source dataset JDT for evaluating feature correlation with ar1. As shown in Fig. 9, CBMMT calculates eight correlation matrices (C_1, C_2,….., C_8) for each of the eight possible combinations of each source chunk with the target dataset ar1 that are (JDT_C1, ar1), (JDT_C2, ar1), (JDT_C3, ar1),……., (JDT_C8, ar1). Each matrix is used to find the number of highly correlated features between two datasets. The threshold for this analysis is set at 0.05. According to the state of the art, the threshold value is decided as 0.05 to cover the maximum number of possible pairs of source and target projects for defect prediction [15]. The threshold value of 0.05, based on the obtained CCVs in the correlation matrix, yields the greatest number of highly associated feature pairs. As a result, the cutoff level is determined based on the estimated CCVs, allowing the DP model to predict defects in maximum number of datasets in the target project. In HDP, this problem is referred to as target prediction coverage (TPC), and the best HCPDP model should obtain the highest TPC for a given prediction combination [36]. Selection of proper threshold to achieve maximum TPC is one of the promising future directions in this domain. As per the analysis, after 30 repeated trials of experiments, the candidate chunk C_4 gives the highest number of highly correlated pairs with ar1. Between JDT and ar1, CBMMT discovered 9 pairs of correlated features.

CIL is now applied to the candidate source chunk C_4 using the novel technique CBA. This algorithm is generating three chunks as the output, each with a balanced number of defective and non-defective instances. Hence, CBMMT and CBA are only used once over the life cycle of the HCPDP-AE model to predict defects between two heterogeneous projects. Thus, the third phase, i.e., metric matching, is carried out to produce the appropriate chunks of training dataset JDT. Finally, the model is trained using a stacking-based ensemble learning (SBEL), with two base models which are trained using K-nearest neighbor (KNN) and random forest (RF) and a meta model that is trained using logistic regression (LR). KNN with higher value of hyper-parameter k will have less impact on DP's performance due to randomness in picking the instances for the training dataset. The findings revealed that k as 18 can control the variance of the model's performance in the experiments. The value of k is finalized on the basis of hit and trial method. Random forest is a classifier that combines a number of decision trees on different subsets of a dataset and averages the results to increase the dataset's predicted accuracy. Instead than relying on a single decision tree, the random forest collects the predictions from each tree and predicts the final output based on the majority votes of predictions. This increases the accuracy of the prediction done by the model. LR is easy to interpret and less inclined to over-fitting. Because CBA and CBMMT use random shuffling of instances in their implementations, the most important factor to consider when choosing classification algorithms is that randomness has a minimal impact on prediction performance. Second, the dataset distribution has a significant role in the selection of classification technique. The ensemble model makes better prediction performance than a single prediction model. SBEL is the most suitable classification algorithm among all ensemble learning strategies when the training data is broken into n distinct fragments. Finally, the performance of the heterogeneous prediction is compared in terms of AUC, recall, accuracy and FS with and without FE as shown in Table 11.

Table 13 Comparative analysis of WPDP-AE and HCPDP-AE

Target dataset	WPDP-AE		HCPDP-AE	
	source dataset	AUC	Source dataset	AUC
ar1	ar1	0.739	JDT	0.717
JDT	JDT	0.702	ar5	0.697
Apache	Apache	0.758	ar1	0.781
ML	ML	0.810	Safe	0.791
ar3	ar3	0.834	EQ	0.788
EQ	EQ	0.789	CM1	0.777
ar4	ar4	0.778	PC1	0.813
PC1	PC1	0.792	LC	0.890
MW1	MW1	0.712	EQ	0.901
CM1	CM1	0.870	EQ	0.803

In all ten prediction variations, HCPDP-AE outperforms HCPDP without FE, as shown in Table 11. The highest and lowest AUC values are 0.901 (HCPDP-P10) and 0.697 (HCPDP-P2), respectively, when the features have been extracted using AE technique. The experimental results under RQ1 and RQ2 show that using AE to obtain strongly discriminated features has a substantial impact on the prediction accuracy of DP, whether it is WPDP or HCPDP. Figure 10 demonstrates a comparison of heterogeneous prediction output with and without FE, taking the average values of all performance parameters into account.

In Table 12, the authors have introduced training time as an extra evaluation criterion to improve the performance comparability. For all prediction combinations, Table 12 compares HCPDP's performance with the original (HCPDP) and reduced (HCPDP-AE) number of features. It can be shown that.

Table 14 HCPDP-AE vs. benchmarked HDP models

Source dataset	Target dataset	TCA +	CCA +	GMOTDP	HCPDP-AE
EQ	ar3	0.5504	0.5846	0.8382	**0.8601**
	ar4	0.5569	0.6462	0.9145	**0.9476**
	ar5	0.5795	0.6923	0.9397	**0.9667**
JDT	ar3	0.5625	0.5692	0.8326	**0.8652**
	ar4	0.5640	0.6462	0.9236	**0.9428**
	ar5	0.6429	0.5692	0.7515	**0.7289**
LC	ar3	0.5982	0.5846	0.8065	**0.8219**
	ar4	0.5530	0.6077	0.9031	**0.9388**
	ar5	0.5661	0.6154	0.9115	**0.9395**
Mean AUC		0.5747	0.6093	0.8690	**0.8901**

HCPDP-AE takes longer time to train than the HCPDP model because the former method comprises the training of a neural network for feature extraction. The average classification time for the HCPDP-AE model is 0.178, which is 37.32% faster than the average classification time for HCPDP.

However, because the HCPDP-AE framework only includes one hidden layer in the encoding–decoding AE model for feature extraction, there is no substantial difference in training and classification time for both approaches. As a result, the execution time cannot be used as an effective assessment parameter to compare the performance of the two models here.

For HCPDP and HCPDP-AE, the mean accuracy, recall, F-Score and AUC are 0.63, 0.67, 0.64, 0.587 and 0.83, 0.84, 0.83 and 0.796, respectively. The target dataset in HCPDP-P5 and HCPDP-P6 is ar3, but the source dataset is different. The prediction accuracy using LC as source dataset is higher than EQ, according to the experimental findings of Table 11. CBMMT has produced 12 and 8 strongly feature pairs for (LC, ar3) and (EQ, ar3), respectively, which explains LC's superior performance due to higher number of correlated features of LC with target dataset ar3. The study shows that the collection of a highly correlated feature pairs set improves heterogeneous predictive performance in a significant fold.

6.3 RQ3. Whether and to What Extent DP Results of HCPDP's Model are Comparable to the Outcomes of WPDP's Model?

The experimental findings for with-in project combinations WPDP-P1 to WPDP-P10 and heterogeneous combinations HCPDP-P1 to HCPDP-P10 can be used to compare the success of the standard method of DP, i.e., WPDP, with heterogeneous prediction. The target projects in these combinations are same. Under WPDP and HCPDP categories, the same as well as different source projects are used to predict defects in these target projects. WPDP's prediction performance is comparable to HCPDP, as shown in Table 13.

Table 15 Output of Wilcoxon signed rank test

Benchmarked model/proposed model	Based metric used to calculate *P*-value		Null hypothesis (H_0)
	Recall	AUC	
TCA + /HCPDP-AE	0.00178	0.0006	Rejected
CCA + /HCPDP-AE	0.00031	0.0012	Rejected
GMOTDP/HCPDP-AE	0.03512	0.0431	Rejected

It is self-evident that a DP model that is trained and tested on the same project makes stronger predictions than a DP model that is trained and tested on different projects. WPDP and HCPDP have almost comparable mean AUCs of 0.778 and 0.796, respectively. Figure 11 presents a comparison of WPDP and HCPDP based on the mean values of all output parameters considered. On the basis of these findings, one may conclude that HCPDP efficiency is comparable to defect prediction within the project with statistical significance. But, when there isn't enough past defect data for the target application to train the DP model, the proposed HCPDP-AE model proves useful.

6.4 RQ4. Compare and Validate the Performance of the Proposed HCPDP-AE Framework with Existing Benchmarked HCPDP Models.

To assess the performance of the proposed algorithm, i.e., HCPDP-AE, three existing benchmarked defect prediction models such as TCA + [31], CCA + [14] and GMOTDP [24] have been considered for the comparative analysis. In heterogeneous defect prediction, TCA + and CCA + are two benchmarked comparative approaches and GMOTDP is very recent work in the SDP domain. One dataset is used as the source project, and three heterogeneous datasets from

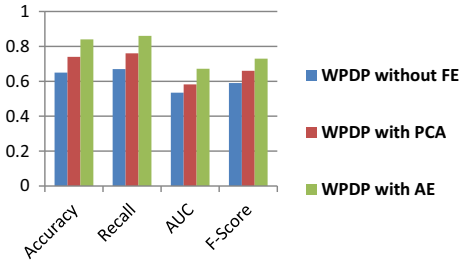


Fig. 8 WPDP's performance comparison with three baselines

another project are used as target projects for heterogeneous prediction throughout the analysis. For evaluating the DP output of all three prediction models, AUC is used as the evaluation index. The AUC estimates the probability that a classification model will distinguish a randomly selected defective instance as being more likely than a randomly selected defect-free example [32]. AUC is more notable in comparison with other performance evaluation metrics (such as accuracy and FS) since it is not influenced by class imbalance issue and irrespective of the prediction threshold, it is used to determine whether an instance should be labeled as a negative instance [9, 33, 34]. According to the literature study [14, 31], CIP is not taken into account in TCA + and CCA + . So, these facts (CIP and prediction threshold) highlight the importance of comparing models performance using AUC, so that uniform pre-conditions (irrespective of CIP) can be achieved for comparative analysis of all four models. Since treating CIP with CBA and selecting the source dataset's chunk for metric matching using CBMMT in HCPDP-AE entail randomness, the average result of 30 repeated trials are counted for each case to mitigate the effect of randomness on the experimental outcomes in training as well as testing of the model. The authors examined these nine prediction pairs for HCPDP_AE only and used experimental results from [24] for the remaining three HDP models.

Table 14 reveals that HCPDP-AE gives remarkable prediction performance than the two classic HDP models (TCA + & CCA +) by 54.88% and 46.09% AUC gain over the mean AUC value of the two models, respectively. There is no greater disparity in mean values of AUC between GMOTDP and HCPDP-AE. Nonetheless, HCPDP-AE outperforms GMOTDP with better DP efficiency, with a 2.43% AUC gain over the latter methodology. The mean AUC for HCPDP-AE is 0.8901, while the mean AUCs for the other three benchmarked models are 0.5747, 0.6093 and 0.8690, respectively, as shown in Table 14.

Figure 12 contrasts the performance of all four HCPDP models on the basis of the mean AUC value, when considering the same nine prediction combinations for the study. This analysis concludes that HCPDP-AE has a stronger prediction effect than the other three models, as shown by the line graph in Fig. 13. So, the performance ranking of the four

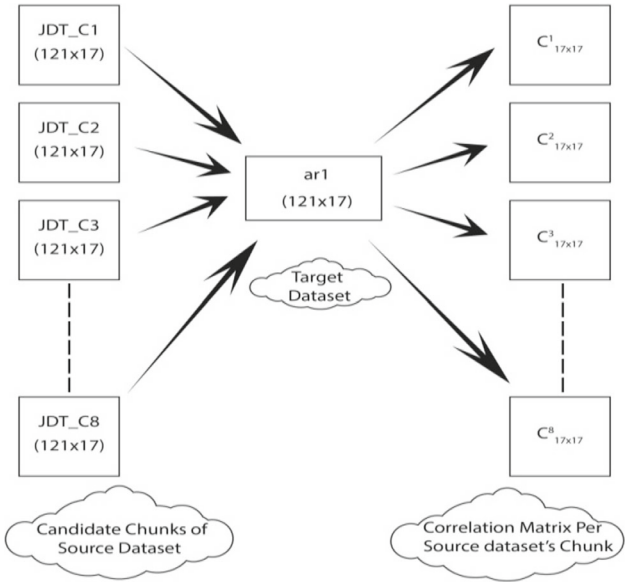


Fig. 9 CBMMT in HCPDP-P1

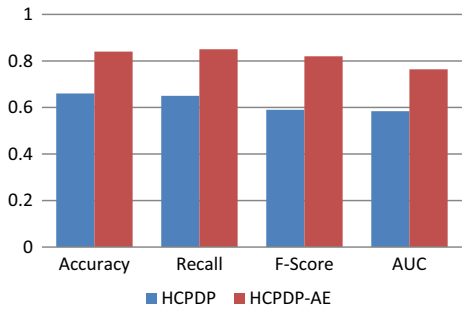


Fig. 10 HCPDP v/s HCPDP-AE

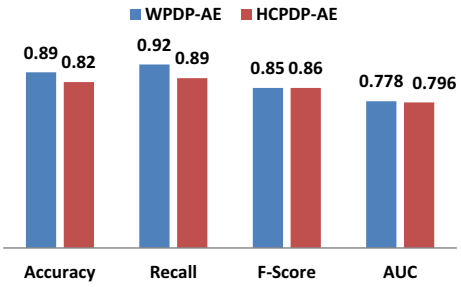


Fig. 11 WPDP-AE v/s HCPDP-AE

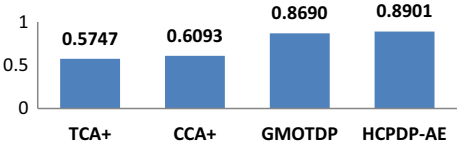


Fig. 12 Comparison between HCPDP-AE with benchmarked models

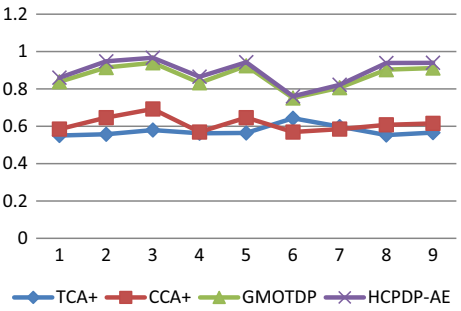


Fig. 13 Plot of AUC values for nine prediction pairs

prediction models in increasing order is given as:-

$$TCA+ < CCA+ < GMOTDP < HCPDP - AE$$

The following are the reasons behind the usefulness and superior performance of proposed model HCPDP-AE:

1. First, it handles CIP better than SMOTE and RUS. It tackles SMOTE's overgeneralization and over-fitting concerns through balancing minority class instances by creating chunks as described in CBA, rather than introducing synthetic instances that duplicate minority class instances. On the other hand, RUS discards majority class occurrences regardless of their importance in prediction of the final outcome. But, CBA does not include the removal of any instance from the dataset. In this manner, CBA addresses the limitations of data re-sampling approaches used to manage CIP.
2. By randomly selecting instances from the original source chunk, traditional HDP techniques [14, 31] create only one training source chunk for the metric matching step. Because there is only one source training chunk, the scope of identifying a better correlation matrix with the target application is limited. Furthermore, the random selection of instances used to build this chunk produces poor results for a while. However, as indicated in Sect. 3, CBMMT gives larger scope by providing numerous source chunks that are used to compute the correlation matrix with the target dataset. The candidates source chunk with the highest number of best CCVs (> 0.05) in the corresponding correlation matrix will be used as the training dataset.
3. Finally, the FE approach AE, which is based on deep learning, aids in increasing the prediction performance of the HCPDP-AE model.

A nonparametric Wilcoxon signed rank test is used to statistically validate the effect of deep learning-based feature extraction on heterogeneous prediction. The test is performed with a P-value of 0.05 (significance level of 5%) to see if there is a significant difference among the performance of proposed model HCPDP-AE and the existing benchmarked models as TCA + , CCA + and GMOTDP. For the analysis, the authors have taken recall and AUC metrics for comparing all the models using 14 heterogeneous prediction combinations that are (EQ → ar3), (EQ → ar4), (EQ → ar5), (JDT → ar3), (JDT → ar4), (JDT → ar5), (LC → ar3), (LC → ar4), (LC → ar5), (JDT → ar1), (ar5 → JDT), (ar1 → Apache), (Safe → ML) and (ML → ar4). AUC and recall are used as assessment measures to execute the test because they have the least or negligible effect on prediction due to the imbalance problem in training datasets [9, 33–35]. Although HCPDP-AE used the robust approach CBA to deal with CIP, CCA + and TCA + do not have this issue addressed in their frameworks [14, 31]. As a result, the authors used these two parameters as evaluation criteria for the test in order to obtain the consistent and unbiased results. If the calculated P-value is less than 0.05, the null hypothesis is rejected. The null and alternate hypotheses are framed as follows:- H_0: The two heterogeneous models are giving same prediction performance. H_1: The two heterogeneous models are giving different prediction performance.

As shown in Table 15, the calculated P-values on the basis of both metrics recall and AUC for all benchmarked models with HCPDP-AE are less than 0.05. That is means that the proposed HCPDP-AE model is performing differently as compared to existing heterogeneous prediction models. Therefore, on the basis of this empirical study, it can be concluded that HCPDP-AE outperforms among all models.

7 Threats to validity

In the proposed feature extraction technique, the number of neurons in the hidden layer is determined by the minimum reconstruction loss. In order to perform metric matching effectively, the CBMMT technique needs at least 15 input features from both projects so that at least 5 correlated feature pairs can be obtained after metric matching to train the DP model effectively. This raises a question about the model's construct validity. The experimental findings can be improved further if the employed classification algorithms are tuned using other optimized options, as the authors used the default options for machine learners in the experiments. This may also be an issue in case of construct validity.

8 Conclusion

HCPDP is a promising area in the SDP domain that enables potentially heterogeneous software project datasets to predict defects on new projects or projects that lack historical defect data to train a DP model. In the paper, the authors proposed

a novel four-phased heterogeneous prediction model using a deep learning-based FE technique called auto-encoder to extract strongly discriminated features that are more relevant to predict expected outcomes. Furthermore, two novel techniques, CBA and CBMMT, are proposed to deal with imbalance problem in training datasets and to evaluate correlation between features of two heterogeneous projects, respectively. For WPDP and HCPDP, the study is able to reduce features by 50.45% and 47.31%, respectively. The experimental findings show that the prediction performance of heterogeneous prediction with and without feature extraction is statistically significant as compared to the respective DP within a project.

The results indicate that using a deep learning method to extract features has a major impact on model prediction accuracy as opposed to using a data-driven FE approach, since a larger number of features contribute to over-fitting and longer processing time to train a model. In addition to this, the authors compared the proposed model's efficiency with three traditional heterogeneous prediction models. HCPDP-AE has been found to be outperformed among all models with the highest mean AUC value of 0.8901.

The future scope of the research is to integrate instance-based filtering and feature extraction in accordance with double pre-processing of datasets. Second, the metric matching process has a huge impact on the prediction performance of any heterogeneous model. As a result, another interesting future direction in this area is to develop a more robust correlation estimation methodology. Future research should also focus on developing an empirical relationship between software defect prediction and predictive maintenance. The authors used a shallow auto-encoder with only one hidden layer in this study. After studying the applicability to a particular problem, one may investigate other variants of AE, such as stacked AE, contractive AE, and de-noising AE as a future work.

References

1. Dajaeger, K.; Verbraken, T.; Baesens, B.: Towards comprehensible software fault prediction models using Bayesian network classifiers. IEEE Trans. Software Eng. **39**(2), 237–257 (2013)
2. D'Ambros, M.; Lanza, M.; Robbes, R.: Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empir. Softw. Eng. **17**(4–5), 531–577 (2012)
3. He, P., Li, B., Ma, Y.: Towards cross-project defect prediction with imbalanced feature sets, CoRR, vol.abs/1411.4228 (2014)
4. Melo, W.L.; Briand, L.C.; Wurst, J.: Assessing the applicability of fault- proneness models across object-oriented software projects. IEEE Trans. Software Eng. **28**, 706–720 (2002)
5. Menzies, T.; Bener, A.B.; Di Stefano, J.S.; Turhan, B.: On the relative value of cross company and within-company data for defect prediction. Empir. Softw. Eng. **14**(5), 540–578 (2009)
6. Camargo Cruz, A. E., Ochimizu, K.: Towards logistic regression models for predicting fault- prone code across software projects. In: Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM), Lake Buena Vista, Florida, USA, pp. 460–463 (2009)
7. Menzies, T., Butcher, A., Cok, D. R., Marcus, A., Zimmermann, T.: Local vs. global models for effort estimation and defect prediction. In: 26th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, Lawrence, KS, USA, pp. 343–351 (2011)
8. Bettenburg, N., Hassan, A. E., Nagappan, M.: Think locally, act globally: Improving defect and effort prediction models. In: 9th IEEE Working Conference on Mining Software Repositories (MSR), IEEE, Zurich, Switzerland, pp. 60–69 (2012)
9. Rahman, F., Devanbu, P., Posnett, D.: 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, pp. 61–65 (2012)
10. Canfora, G., De Lucia, A., Oliveto, R., Panichella, A., Di Penta, M., Panichella, S.: Multi objective cross-project defect prediction. In: IEEE Sixth International Conference on Verification and Validation in Software Testing, IEEE, Luxembourg, ISSN 2159–4848 (2013)
11. Gao, K.; Khoshgoftaar, T.M.; Zhang, H.; Seliya, N.: Choosing software metrics for defect prediction: an investigation on feature selection techniques. Softw Pract. Exper. **41**(5), 579–606 (2011)
12. Wang, T.; Zhang, Z.; Jing, X.; Zhang, L.: Multiple kernel ensemble learning for software defect prediction. Autom. Softw. Eng. **23**(4), 1–22 (2015)
13. He, J.Y.; Meng, Z.P.; Chen, X.; Wang, Z.; Fan, X.Y.: Semi supervised ensemble learning approach for cross-project defect prediction. Journal of Software Engineering. **28**(6), 1455–1473 (2017)
14. Dong, X., Jing, X., Qi, F., Wu, F., Xu, B.: Heterogeneous cross company defect prediction by unified metric representation and CCA-based transfer learning. In: Proceedings of 10th Joint Meeting on Foundations of Software Engineering, ACM, New York, NY, USA, pp. 496–507 (2015)
15. Fu, W., Kim, S., Menzies, T., Nam, J., Tan, L.: Heterogeneous defect prediction. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE, ACM, New York, NY, USA, pp. 508–519 (2015)
16. Ryu, D.; Jang, J.-I.; Baik, J.: A transfer cost-sensitive boosting approach for cross-project defect prediction. Software Qual. J. **25**(1), 1–38 (2015)
17. Ryu, D.; Baik, J.: Effective multi-objective naive Bayes learning for cross-project defect prediction. Appl. Soft Comput. **49**, 1062–1077 (2016)
18. X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu: Heterogeneous cross company defect prediction by unified metric representation and CCA-based transfer learning. In: Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE, pp. 496–507 (2015)
19. C. Ni, W. Liu, Q. Gu, X. Chen, and D. Chen: FeSCH: A Feature Selection Method using Clusters of Hybrid-data for Cross-Project Defect Prediction. In: Proceedings of the 41st IEEE Annual Computer Software and Applications Conference, COMPSAC, pp. 51–56 (2017)
20. Li, Y., Huang, Z., Wang, Y., Fang, B.: Evaluating data filter on cross-project defect prediction: comparison and improvements. In: IEEE Access 5, ISSN 25646–25656 (2017)
21. Xu, Z., Yuan, P., Zhang, T., Tang, Y., Li, S., Xia, Z.: HDA: Cross project defect prediction via heterogeneous domain adaptation with dictionary learning. In: IEEE Access 6, 57597–57613 (2018)
22. Lee, S.P., and Felix, E.A.: Predicting the number of defects in a new software version. PloS ONE. 15(3) 2020

23. Majd, A., Vahidi-Asl, M., Khalilian, A., Poorsarvi-Tehrani, P., and Haghighi, H.: SLDeep: Statement-level software defect prediction using deep-learning model on static code features. Expert Syst. Appl. 14(7) 2020
24. Jiang, K.; Zhang, Y.; Wu, H.; Wang, A.; Iwahori, Y.: Heterogeneous Defect Prediction Based on Transfer Learning to Handle Extreme Imbalance. Appl. Sci. (2020). https://doi.org/10.3390/app10010396
25. Marqués, A.; García, V.; Sánchez, J.: On the suitability of resampling techniques for the class imbalance problem in credit scoring. J. Oper. Res. Soc. **64**, 1060–1070 (2013). https://doi.org/10.1057/jors.2012.120
26. Vashisht, R., Rizvi, S.A.M.: Feature extraction to heterogeneous cross project defect prediction. In: 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1221–1225 (2020). https://doi.org/10.1109/ICRITO48877.2020.9197799
27. Fan, C.; Sun, Y.; Zhao, Y.; Song, M.; Wang, J.: Deep learning-based feature engineering methods for improved building energy prediction. Appl. Energy, Elsevier. **240**(C), 35–45 (2019)
28. Stacey, R.: Deep learning: which loss and activation functions should i use? [Online] Available at: https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8(2018) (Accessed: 27 July 2018)
29. Zhu, K.; Zhang, N.; Ying, S.; Wang, X.: Within-project and cross-project software defect prediction based on improved transfer naive bayes algorithm. Comput. Mater. Continua. **63**(2), 891–910 (2020)
30. Maheshwary, S.; Kaul, A.; Pudi, V.: Data Driven Feature Learning (2017)
31. Nam, J.; Pan, S.J.; Kim, S.: Transfer defect learning. In: Proceedings of the 2013 International Conference on Software Engineering, Piscataway, NJ, USA: IEEE Press, pp. 382–391 (2013)
32. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
33. Giger, E., D'Ambros, M., Pinzger, M. Gall, H.C.: Method level bug prediction. In: Proceedings of the 6th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM2012, 171–180 swe. (2012)
34. Song, Q.; Jia, Z.; Shepperd, M.; Ying, S.; Liu, J.: A general software defect-proneness prediction framework. IEEE Trans. Software Eng. **37**(3), 356–370 (2011)
35. Amalia, L., Alejandro, C., Alejandro, M., Ana de las, H.: The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recogn. **91**, 216–231, ISSN 0031–3203, (2019). https://doi.org/10.1016/j.patcog.2019.02.023
36. Vashisht, R.; Rizvi, S.A.: Estimation of target defect prediction coverage in heterogeneous cross software projects. Int. J. Inf. Syst. Model. Design (IJISMD) **12**(1), 73–93 (2021). https://doi.org/10.4018/IJISMD.2021010104
37. Vashisht, R.; Rizvi, S. A.: Class imbalance learning to heterogeneous cross software projects defect prediction. Int. J. Software Innov. (IJSI), **10**(2), Article 4 (2021)
38. D'Ambros, M.; Lanza, M.; Robbes, R.: An extensive comparison of bug prediction approaches. In: 7th IEEE Working Conference on Mining Software Repositories (MSR), pages 31–41 (2010)
39. Boetticher, G.; Menzies, T.; OstrandJ, T.: The PROMISE repository of empirical software engineering data (2007). http://promisedata.org/repository
40. Wu, J.; Wu, Y.; Niu, N.; Zhou, M.: MHCPDP: multi-source heterogeneous cross-project defect prediction via multi-source transfer learning and autoencoder. Software Qual. J. **29**, 1–26 (2021). https://doi.org/10.1007/s11219-021-09553-2

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.