Cross Project Software Defect Prediction Using Machine Learning: A Review

Muhammad Salman Saeed, Muhammad Saleem

Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan
National College of Business Administration & Economics, Lahore, Pakistan
*Corresponding Author: Salman Saeed. Email: salman.saeed403@gmail.com

Abstract: Software defect prediction is a crucial area of study focused on enhancing software quality and cutting down on software upkeep expenses. Cross Project Defect Prediction (CPDP) is a method meant to use information from different source projects to spot software issues in a specific project. CPDP comes in handy when the project being analyzed lacks enough or any data about defects for creating a dependable defect prediction model. Machine learning that is a part of artificial intelligence learns from data and then makes forecasts or choices. Machine learning (ML) is a key component of CPDP because it can learn from heterogeneous and imbalanced data sources. However, there are many challenges and open issues in applying machine learning to CPDP, such as data selection, feature extraction, model selection, evaluation metrics, and transfer learning. In this study, we provide a complete review of existing literature from 2018 to 2023 on Defect Prediction using Machine Learning, covering the main methods, applications, and limitations. We also use ML to identify current research gaps and future directions for CPDP. This paper will serve as a useful reference for researchers interested in using ML for CPDP.

Keywords: *Cross Project Software Defect Prediction, Machine Learning, Review Paper*

1 Introduction

Software Defect Prediction (SDP) involves finding and pinpointing parts of software that might have problems. It assists software developers and testers in using their resources and efforts more wisely, ultimately enhancing the quality and trustworthiness of software products [1], [2]. Usually, SDPs rely on data of past project from the same project (known as within-project defect prediction or WPDP) to make models for predicting defects using machine learning (ML) methods. However, sometimes there's not enough or no defect data for the same project, especially for new or ongoing projects. This makes it challenging to apply and accurately use the WPDP model.

Cross Project Defect Prediction (CPDP) offers a better solution of this problem. It uses defect data from different projects to forecast defects in a new project, helping to overcome the lack of data issue and providing predictions for items with little or no defect history [3]. CPDP benefits from the variety of data available from various source projects, which capture different aspects of software defects. However, CPDP also faces challenges and unanswered questions, like picking suitable and trustworthy source items, extracting and aligning features from different data sources, selecting the right machine learning models and settings, evaluating and comparing CPDP models, and transferring knowledge between different domains and scenarios [4, 84].

Lately, the blending of software engineering and machine learning has brought hopeful solutions to persistent problems. By leveraging rich historical project data, machine learning techniques offer the potential to transform software engineering from a reactive to a proactive process. By analyzing patterns, relationships, and factors that have contributed to past defects, these techniques can identify areas of code that are more likely to develop defects in future projects. Researchers have proposed some machine learning techniques [5-13, 85-89] to improve the software engineering process. This shift towards predictive defect management is expected to not only improve software quality but also optimize resource allocation through more rigorous testing and review of defect-prone parts [92, 93].

In this research, we provide a complete summary of the current body of knowledge regarding CPDP with machine learning. This review encompasses key techniques, practical uses, and constraints. We also identify current research gaps and future directions for using ML for cross-project software defect prediction. By providing a comprehensive overview of the existing literature on CPDP using ML, this research plans not only to promote a deeper understanding of the field, but also to provide valuable insights to researchers, practitioners, and software developers interested in implementing CPDP techniques in practice. To conduct this review, detailed guidelines were obtained from some previous literature reviews [14-18]. By identifying the strengths and limitations of current approaches, this review can also catalyze further research and development, ultimately driving advances in defect prediction across projects and contributing to overall improvements in software quality and reliability.

A study [19] examines the performance of re-sampling methods in tackling class imbalance issues within the SDP process. This imbalance problem can impact accuracy and trustworthiness of models. The research introduces a framework that enhances SDP model performance through ensemble learning and feature selection methods. Six public datasets from NASA are used to evaluate this framework, and it's compared against ten supervised classification techniques. The study asserts that this framework surpasses all other techniques in F-measures, accuracy, MCC, and ROC.

In one study [20], a feature selection process is proposed for CPDP, where they aim to combine CFS (correlation-based feature selection) and the Relief algorithm (Relief). In another study [21], a novel approach called the Ensemble Oversampling Model (EOM) is introduced. EOM combines multiple oversampling techniques to create a balanced dataset and then employs an ensemble classifier to predict defects. This research assesses EOM's performance using 12 datasets of software engineering from the broadly used PROMISE repository. The paper compares EOM with four standard classification techniques: Support Vector Machine, Naive Bayes, Random Forests (RF), and Logistic Regressions. Two evaluation parameters are used: the accuracy rate (ACC), which measures overall prediction correctness, and false negative rate (FNR), it point to the percentage of defective components missed by the classifier.

In another study [22], a new approach is proposed that combines deep learning and ensemble learning to increase the accuracy and robustness of the models of software defect prediction (SDP). The research begins by introducing the background and principles of SDP and reviews previous efforts in this area using both deep learning and traditional machine learning methods. The study then outlines the proposed framework, which involves four key steps: feature extraction, data preprocessing, model training, and model fusion. Detailed explanations, algorithm pseudo-code, and network architecture diagrams are provided for each step.

In a separate paper [23], the focus is on using Neural Networks-based ensemble techniques to calculate defective software components. This paper finds to improve the accuracy and reliability of software defect prediction models by using the multi-layer perceptron as the foundational classifier and combining it with ensemble techniques such as bagging and boosting. The study evaluates this framework using public NASA datasets and compares it to ten supervised classification techniques, including

ensemble learning techniques like bagging and boosting. According to the paper, this framework does better than all other techniques in accuracy, MCC, F-measure, and ROC.

Another new approach to CPDP uses data from different projects to calculate software defects in one project. The method, called WCM-WTrA, method merges feature selection techniques and distance weighted instance transfer to reduce variance between items and improve prediction accuracy. The paper also extends the method to handle multiple source terms and calls it Multi-WCM-WTrA [24]. In a different study [25], a new approach is presented for CPDP, which involves using training data from a project to forecast software bugs in another project. This method, known as TFS, employs transformation and feature selection techniques to lessen the disparities between the training and test projects. The transformed and selected data from the source projects are then used to forecast bugs in the test projects using a random forest classifier. The study concludes that TFS generally performs better than other methods and shows significant enhancements over contemporary CPDP models.

In a study [26], they propose a classification approach for SDP that relies on feature selection techniques. This method is designed to increase accuracy and dependability of software quality evaluation. The framework comprises three main components: dataset and feature selection, the classification stage, and result analysis. They apply this method in two ways: one with feature selection and one without. The research conducts experiments using 12 publicly available datasets from NASA and compares the outcome of their framework with other commonly used classification methods. According to the paper, their framework outperforms some of the commonly used classification techniques in certain datasets.

Another approach to using cross-project data to identify where crashing faults reside is proposed, by predicting whether the fault that caused the software to crash is inside or outside the stack trace of a crash report. The method, called FSE, uses a cross-project framework of feature selection and embedding to deal with data scarcity in this task. We then use the selected embedded source item data using a random forest classifier to predict crash fault locations in the target item. This paper evaluates the method on 7 real projects and compares it with 25 other methods. The paper finds that FSE outperforms other methods in most cases and shows superior performance in identifying crash fault residency [27].

A study [28] introduced a novel EL (ensemble learning) model for SDP, using kNN, GLMNet (generalized linear models with elastic network regularization) and random forests as the underlying learner linear Discriminant analysis (LDA). The authors claim that the EL technique can decrease inconsistency and bias between source and target datasets and improve prediction accuracy and performance. The authors conclude that ensemble machine learning models are capable and effective for SDP. A paper [29] proposes a method for SDP using ensemble learning (EL) and feature selection (FS) techniques. This method uses six NASA datasets, applies different search methods for attribute selection, and uses EL techniques to build defect prediction models. The framework is evaluated using various performance metrics and compared with other classification techniques. The paper shows that the framework increases accuracy and reliability of SDP models.

In a different study [30], they established a data transformation method for CPDP founded on Generative Adversarial Networks (GANs). Their research suggests that GAN-based techniques can decrease the differences in data between training and testing projects, leading to enhanced defect prediction performance. The authors adopt a GAN-based approach, selecting the labelled dataset as real data and the target dataset as bogus data. The discriminator measures the domain version through a loss function, while the generator tries to adapt the target dataset to the source item domain. Then, the authors apply a Naive Bayesian classifier to classify the faulty modules.

There is another novel cost-conscious bagging ensemble (CCBE) method for CPDP, which uses re-sampling techniques to construct cost-aware classifiers and ensemble techniques to capture better

predictable sex. The generalization property addresses the class imbalance in each classifier. The authors claim that the CCBE method can automatically identify cost values during model training, improving predictive performance and practicality [31].

In another study [32], a software defect prediction framework is introduced, which employs multi-filter attribute selection techniques in conjunction with the multi layered perceptron classifier. They utilized 12 datasets from NASA, applies various filtering methods for feature selection, and employs MLP to construct a defect prediction model. This framework's performance is assessed using four performance metrics and is compared to other classification techniques. The study demonstrates that this framework enhances accuracy and reliability of SDP models.

The paper [33] proposes a Multi-Adaptation and Kernel Specification (MAN2)-based CPDP (Project Defect Prediction) method, which tends to reduce domain differences and data noise between training and test projects. The authors show that the MAN2 method can learn low-rank sparse representations of source data and domain-invariant feature spaces of target data by using kernel-norm regularization and multiple adaptation techniques. In paper [34], an intelligent system is introduced to detect faulty modules in software using a two-step predictive method. The initial step involves applying the three machine learning techniques to given data: ANN, NB, and DT. The subsequent step combines the classifier's accuracy with a fuzzy logic-based system to achieve improved performance. They utilize datasets from well-known NASA repositories. According to the paper, this system surpasses other techniques, including basic classifiers and contemporary ensembles, in terms of performance.

There is another method proposed in a study [35] which is called KPWE, which combines two techniques: Weighted Extreme Learning Machine (WELM) and Kernel-Principal Component Analysis (KPCA). WELM is a fast and robust learning algorithm that can alleviate the problem of class imbalance by allocating different weights based on the proportion of different classes. KPCA is a nonlinear dimensionality reduction method that can capture the nonlinear relationship between data characteristics. There is another CPDP algorithm based on transfer learning (TL), which aims to learn the common attribute space and domain-specific attribute space of source items and target items by using a two-stage TL framework. Studies have shown that TL-based algorithms can reduce domain differences and data noise between source and target items, and increase the performance and stability of defect prediction [36].

The authors of the paper [37] evaluated 10 supervised machine learning techniques on four NASA datasets containing software metrics and defect information. They compare these techniques using F-measure, Accuracy and MCC as performance metrics. They also use attribute selection and re-sampling methods to shrink the dimensionality and imbalance of the dataset. They found that multilayer perceptron’s and SVMs were the most effective techniques on the NASA dataset, and that feature selection and re-sampling techniques can improve performance of ML techniques.

In reference to study [38], a recommender system is developed for selecting data methods in CPDP. The system's goal is to suggest the most suitable method for selecting training data from training projects for a given test project. The authors report that this recommender system suggests the best CPDP data selection method and performs better than the baseline method they considered.

As for study [39], it delves into how different techniques for reducing features can affect performance and variance of various prediction models. The research compares attribute reduction methods like Principal Component Analysis, Linear Discriminant Analysis, and Auto encoders with feature selection methods like as Information Gain, Chi-Square, and Relief. This study evaluates these methods using six defect prediction models, including support vector machines, random forests, and

means clustering. The research highlights that feature elimination methods can often enhance the performance of models and reduce variance, particularly for unsupervised models and large datasets.

In the paper [40], they present a technique for SDP that integrates feature selection, data fusion, and ensemble machine learning methods. This framework utilizes a range of ML models that consists of SVM, DT, and NB, in combination with ensemble machine learning methods like voting, bagging, and stacking, to predict faulty software modules. Furthermore, fuzzy logic techniques are applied to combine the results from ensemble machine learning methods. The framework's performance is evaluated using four real-world software datasets and exhibits better results when compared to specific existing methods.

One study [41] is about a novel system that uses ML methods to forecast software defects. This system extracts various features from the given data and then applies three different classifiers: Naive Bayes, Artificial Neural Networks, and Decision Trees. The system then combines the results of these classifiers using fuzzy logic-based fusion techniques to obtain a final prediction. The paper claims the system achieves greater accuracy and precision than modern methods. The paper also evaluates the system on five datasets collected from NASA projects and compares it to other existing methods. A study [42] evaluated the performance of CNN (convolutional neural networks) in multi-category CPDP, aiming to classify software components into four defect severity levels: critical, major, minor, and trivial. There is also a study discovering the viability of applying federated learning (FL) to Cross Projects, aiming to preserve the privacy of source items by training a global model without sharing raw data [43].

An article [44] presents a new deep forest model for SDP, a technique for identifying defective software components before they cause failures. Deep forest models consist of multiple levels of decision trees that can learn complex non-linear features from software metrics data. The model also uses a waterfall structure that can dynamically adjust the number of tiers based on data complexity and predictive performance. In method [45], a novel approach is introduced to automatically identify code characteristics linked to software defects by using n-gram abstract syntax tree (AST). AST n-grams are series of nodes within the AST representation of source code, which capture both the semantic and syntactic configuration of the code. This research compares the performance of AST n-grams with other methods for extracting code features, such as bag-of-words, bag-of-tokens, and the Halstead metric, across six open-source projects. The article demonstrates that AST n-grams achieve higher precision and recall in software defect prediction compared to other methods and are competent to identify the most related code features for each item.

In study [46], the authors introduce a different approach for CPDP, which involves leveraging data from multiple sources to forecast software bugs in a specific project. This method, known as MFTCPDP, employs a technique called manifold feature transformation to reduce the disparities in data sharing between the training and test projects. Manifold feature transformation entails converting the novel attribute space of an item into a low-dimensional diverse space that retains the essential structure of the data. Subsequently, the transformed data from source items is utilized to predict defects in target items using a Naive Bayesian predictive model.

A paper [47] proposes another method for CPDP; it utilizes the data from many projects to calculate software faults in one project. The method, called NN Filter, uses a data resampling method to deal with CIP in defect prediction datasets. It evaluates the effect of five oversampling methods (SMOTE, MAHAKIL, Borderline-SMOTE, ADASYN and random oversampling) and three under-sampling methods on different models. The paper compares the results to those of methods without data resampling. A paper proposes another approach to CPDP, a task that takes data from source project to forecast software bugs in test project. The method, called CFIW-TNB, uses correlated features and instance weight transfer learning to shrink the difference between train and test projects. Then, this paper uses the

selected and weighted source item data to predict defects in target items using a Naive Bayesian classifier with a tree-augmented network structure [48].

There is another paper [49] that aims to improve CPDP, a technique that takes data from training projects to recognize faulty software components. This paper focuses on the role of prediction metrics, which are numerical factor of software quality, diversity, and complexity, in the CPDP model. This paper suggests a metric selection technique based on metric diversity; it measures how different metrics capture different aspects of software characteristics. This paper evaluates the given technique on different projects and compares it with existing metric selection methods and modelling techniques. The paper reports that the given technique finds better accuracy and stability than baseline techniques and that metric diversity is an important factor in CPDP performance.

The study [50] classifies existing defect prediction techniques into four areas: manipulated data, prediction algorithms based on machine learning, empirical research and effort-aware prediction. This paper also discusses some research challenges and opportunities for future defect prediction, such as dealing with unbalanced and noisy data, integrating domain knowledge, evaluating prediction models, and applying defect prediction in practice. There is a research article [51] that explores how different methods of selecting features affect the accuracy of predicting software defects in different projects. This study introduces a new attribute selection that is a method based on the correlation between features and defects and compares it with six existing methods using 12 datasets from different software domains. The paper reports that their method outperforms other methods on evaluation metrics: accuracy, recall, precision, and F1-score. This paper also discusses the implications and limitations of their findings and suggests some directions for future work.

There is a paper [52] that aims to investigate how different hyper-parameters of a machine learning classifier influence the CPDP model performance for identifying software defects in projects with no or insufficient historical data. It compares five different approaches to instance-based selection, which are techniques for selecting subsets of data from source items that are most relevant to a target item. It used eight items in two datasets as evaluation objects and utilized the AUC as a metric. The paper found that hyper-parameter optimization had a significant impact on four of the five methods, with the most influential hyper-parameters related to SVM and KNN classifiers. It also analyzes the cost of hyper parameter optimization and concludes that it lies in a satisfactory range. A paper [53] presents a novel deep-learning framework for predicting software defects in cloud environments, which are complex and dynamic systems. The framework uses various deep learning methods i.e. RNN, LSTM and CNN networks to learn the characteristics and patterns of software indicators and defects. The framework is evaluated on four datasets from cloud software projects, showing better performance than several existing methods. This paper aims to enhance the value and consistency of cloud software development.

In study [54], an SDP model about LASSO-SVM is proposed, combining the LASSO method with the SVM algorithm. This model aims to enhance both the accuracy and speed of SDP. It achieves dimensionality reduction of datasets using LASSO and employs SVM for nonlinear classification. They used cross validation to adjust the SVM constraint. The model's performance is assessed using two real datasets from NASA and PROMISE, and it is compared to other SDP classifiers i.e. Logistic Regression, SVM, Decision Trees, NB, and RF. The results demonstrate that the model achieves better precision, recall and f-score rates on the two datasets. Additionally, the model outperforms other SDP models in terms of prediction speed.

Another paper [55] proposes a technique to handle the imbalanced class distribution in CPDP using an ensemble under-sampling technique. This technique combines multiple classifiers with different under-sampling rates to balance classes and improve prediction accuracy. The paper evaluates the given technique on 10 real datasets and compares it with four methods that exist already: Naive Bayes, Random

Forest, SMOTE and TCA+. It reports that the method does better than other methods in precision, recall, F-measure, and AUC scores. The paper also analyzes the impact of different under sampling rates and ensemble techniques on prediction performance. Another paper [56] presents a new method for predicting software defects in different projects using data from other projects. The method combines weighted nearest neighbor and grey relational analysis to deal with missing values in the data. The paper also compares different methods for selecting the most significant features for defect forecasting. The paper tested the method on seven datasets and reported promising results.

There is another oversampling method proposed [57] called SPIDER3, which represents synthetic populations of imbalanced data using evolutionary rules. SPIDER3 uses genetic programming to generate synthetic instances that are similar to, but not identical to, minority class instances. SPIDER3 also considers the distribution of most class instances and avoids overlapping with them. The paper compares SPIDER3 results with other oversampling methods such as Borderline-SMOTE, SMOTE, MWMOTE and ADASYN, as well as cost-sensitive students such as CSB1, AdaCost, and CSB2. The paper uses various evaluation parameters such as accuracy, precision, recall, F-measure, G-mean, AUC-ROC and so on. An article [58] presents a framework for SDP using attention-based recurrent neural networks (ARNNs). The paper claims that the ARNN can automatically learn the semantic and syntactic features from an Abstract Syntax Tree (AST) and use a method to produce features important for perfect fault prediction. This article evaluates the framework on some Java projects and compares it to modern techniques.

In article [59], an ensemble ranking method based on voting is suggested, it combines the predictions of three core "students" - Adaboost, RF, and NB. The paper also employs wrapper-based attribute selection techniques to decrease data dimensionality and select the most relevant features for each of these base models. This method is evaluated using six NASA datasets and is compared to existing methods like LR, SVM, DT, KNN, and bagging. Many metrics i.e. precision, accuracy, recall, F1-score, G-mean, and AUC are used for evaluation.

In paper [60], an ensemble learning approach for SDP using various ML methods is presented. The aim is to increase accuracy and reliability of defect prediction by leveraging the strengths of different models of machine learning. This study uses four base models: KNN, Decision Tree, SVM, and Naive Bayes, along with Bagging and Boosting. And performance of this approach is assessed on six publicly available PROMISE datasets. The article compares the results of the ensemble method with those of the base models and other techniques.

In paper [61], a new fault prediction technique based on Deep Belief Network (DBN) and L1 regularization optimization is proposed. DBN is an artificial neural network that can learn multifaceted nonlinear model from data, and L1 regularization is used to prevent overfitting and enhance generalization by penalizing large weights in the network. The article claims that this model surpasses available techniques in accuracy, recall, precision, and F1-score. Experimental results from four public datasets are presented to support this claim, and the given model is compared with other techniques i.e. logistic regression, DT, RF, and SVM.

In study [62], the authors compare the output of different algorithms for SDP. These task identifies defective modules before testing or implementation. This paper utilizes five PROMISE datasets, each containing software parameters and defect labels for different software projects. Cross-validation is performed ten times, and the accuracy, recall, precision, and F-scores on each dataset are evaluated. The algorithms tested on DT, RF, SVMs, nearest neighbors, NB, logistic regressions, artificial neural networks, extreme gradient boosting, adaptive boosting, and bagging.

Another paper [63] is about to enhance the accuracy of SDP models by applying different machine learning techniques and optimization methods. This study utilizes four datasets from the Promise repository, which contain information about software modules and their defects. Six machine-learning techniques are applied to build predictive models: SVM, LR, KNN, decision trees, random forests, and ANN. Additionally, four optimization methods are employed to improve model performance.

In article [64], a seven-ensemble ML model is proposed, combining LGBM, Cat boost, boosted Cat boost, XgBoost, bagged Logistic Regression, boosted XgBoost and boosted LGBM. The study evaluates these models using six datasets from the PROMISE archive, assessing some parameters of performance i.e. accuracy, AUC, precision, F1-score, recall, and MCC. The article suggests that the ensemble Cat-boost method outperforms others on three defect datasets while reducing overfitting and training time.

2 Methodology

The purpose of this study is to offer a general examination of the latest techniques and approaches in CPDP using machine learning. CPDP entails constructing a predictive model on a source item and using it on a target item with distinct characteristics.

A study [65] addressed the problem of HCPDP (heterogeneous CPDP), it forecasts faults in test projects using data from different source projects. This paper proposes two novel methods based on optimal transfer theory, which can measure the similarity and transferability between train and test distributions, and reduce the negative impact of distribution mismatch. The first method, called OTHCPDP, uses optimal transfers to align train and test data distributions and then applies a classifier to predict defects in target items. The second method, called OT-HCPDP+, extends OT-HCPDP by combining attribute selection and EL (ensemble learning) techniques to further enhance predictive performance and robustness. This paper evaluates the proposed method on the real datasets of AEM, JIRA, NASA and PROMISE software projects and compares it with some modern HCPDP methods. The paper reports that the given method gains better or comparable results as compared to already available methods in accuracy, F-measure, G-mean, and AUC. This paper also conducts statistical tests and sensitivity analysis to authenticate the efficiency and stability of the given technique.

There is another study [66] that proposed a Genetic Algorithm Feature Selection (GAFS)-based CPDP method aiming to perk up the output and robustness of CPDP to find the best subset of features for each target item. The paper introduces the concept of CPDP, which leverages data from other related projects to construct a model for defect prediction for new projects that have little or no historical data. This paper also reviews existing CPDP methods and their limitations, such as distribution mismatch, class imbalance, and feature redundancy. The paper proposes the GAFS method, which consists of two steps: attribute selection and ensemble training. In the attribute selection phase, a global search adaptive attribute selection technique based on GA (a genetic algorithm) is proposed; it uses the comprehensive results of the candidate attribute subsets on the test data to drift the optimal attribute subset. In the second stage, this paper utilizes the Easy Ensemble technique to increase the CIP (class imbalance problem), and constructs multiple NB (naive Bayesian) classifiers, and then builds the model through ensemble learning. This paper evaluates the GAFS method on the very popular real-world datasets in the AEEEM and PROMISE data repositories and compares it with five modern CPDP techniques in MCC and F-scores. The paper reports that GAFS can attain equivalent results than available techniques, and can significantly improve the average F1-score and MCC. The paper also conducted statistical tests and sensitivity analysis to verify the validity and stability of GAFS.

In paper [67], various approaches to predict software defects are evaluated in two scenarios: within the project and across-projects. Within-project prediction involves using data from same project to train and test the model, while CPDP uses data from different projects. The paper compares four types of

defect prediction methods: statistical, machine learning, meta-learning, and hybrid. Datasets from PROMISE and AEEEM repositories are employed, and several metrics i.e. precision, accuracy, recall, F1-scores, and AUC-ROC are applied. The study found that a hybrid technique that integrates statistical and ML methods outperformed other methods in both scenarios. Additionally, the paper highlights practical challenges and limitations in defect prediction, including data quality, data imbalance, feature selection, and model interpretation. It suggests future research directions, such as enhancing data preprocessing, exploring new features, and developing interpretable models.

In paper [68], the focus is on the effect of important feature selection by hybrid methods in CPDP for multiclass datasets. The study suggests a hybrid method for feature selection based on RF and recursive attribute exclusion cross validation, it selects a small number of meaningful and relevant features for defect prediction. A CNN serves as the classifier for predicting defects across projects, with SoftMax that is the last layer. The given technique is assessed on various PROMISE datasets, achieving an average prediction accuracy of 78% with AUC. The study concludes that hybrid feature selection significantly influences defect prediction accuracy across different datasets in cross-project environments.

A study [69] suggested a two-stage framework for CPDP, a technique that uses data from different projects to identify defective software modules. The first stage of the framework is indicator selection, which aims to select the most relevant and informative indicator from a large number of candidate indicators for defect prediction. This paper adopts a hybrid technique for feature selection that is based on RF and recursive feature removal cross validation, that can sort indicators according to their importance and eliminate redundant and irrelevant indicators. The second stage of the framework is the balancing method, which aims to manage the class imbalance problem (CIP) in defect prediction, it occurs when the number of defective components is very small in number to the non-defective components. The paper uses a hybrid ensemble method based on SMOTE and bagging, which can generate a synthetic model for minority classes and decrease the variance of the classifier. This paper evaluates the proposed framework for the PROMISE repository dataset. It also compares the output of the framework with six available CPDP techniques in terms of the F1 score. The paper reports that the given framework achieves better or comparable results than existing techniques and significantly improves the average value of all evaluation metrics.

In paper [70], a novel method for predicting software failures in target projects using data from different source projects is proposed. This approach leverages artificial intelligence techniques like deep learning, genetic algorithms, and fuzzy logic. A deep neural network (DNN) is employed as a classifier to predict the failure likelihood of software components, trained on data from multiple source projects. Genetic algorithms (GA) are utilized to optimize the hyper-parameters of DNN, i.e. neurons, activation functions and hidden layers. Fuzzy logic is applied to assess the similarity and transferability between train and test projects, aiding in selecting the most suitable source project for DNN training. Pythagorean fuzzy sets (PFS) are used to represent uncertainty and ambiguity in software metrics and fault labels. The method's effectiveness is demonstrated using data from the PROMISE repository.

Paper [71] introduces a technique to deal with noise and class imbalance in heterogeneous CPDP. CPDP involves using data from different projects to identify software modules that may contain defects. Noise refers to erroneous or irrelevant data that can impact prediction accuracy, while class imbalance occurs when the total defective and non-defective modules is unequal, potentially biasing the prediction model. The paper introduces the Block Balancing Algorithm (CBA) to handle class imbalance and evaluates four different classification algorithms to deal with noise. The method is evaluated using AEEEM, SOFTLAB, and Relink datasets. The study finds that noise and class imbalance significantly affect defect prediction accuracy, and the proposed technique and classification algorithm improve prediction results. Various performance parameters, including accuracy, F1 score, confusion matrix, and AUC, are employed.

In study [72], a technique for selecting the best features from multi-class data for CPDP is suggested. For feature selection, it utilizes a search-based optimizer called MOEA/D (corrosion-based multi objective evolutionary algorithm). The given technique is compared with other methods of feature selection like Relief, Information Gain, and CFS on multiple datasets from the PROMISE repository. Different classifiers, including NB, SVM, KNN, and RF, are used to assess the output of the techniques of attribute selection. The study claims significant improvements in the F1 metric and demonstrates that the proposed method outperforms others on most datasets.

In paper [73], the authors review existing literature on Software Defect Prediction (SDP), software testing, and machine learning techniques. They propose a new method CPDP using deep learning. A DNN model is employed to learn the features and patterns from training project data and implement the learned methods to test project data. This approach is compared to other ML models like NB, Random Forest, SVM, and KNNs on multiple datasets from the PROMISE repository. The study claims better results in precision, F-score, accuracy, and recall, demonstrating the effectiveness of the DNN model in addressing data imbalance and distribution mismatch in CPDP.

Paper [74] presents a novel deep learning model for CPDP. This technique uses data from different projects to estimate defect-prone modules in a software product. The study employs a method to capture semantic and structural information from both training and test projects. Additionally, a transfer learning strategy is applied to adapt the model to different domains. The given model is evaluated on PROMISE datasets and compared to other modern CPDP methods like Transfer Naive Bayes, Transfer Component Analysis, and Deep Transfer Learning. The study reports excellent results in F-scores and highlights the effectiveness of the self-attention mechanism in addressing data imbalance and feature heterogeneity in CPDP.

In paper [75], a method for SDP is presented, utilizing attribute selection and variant-based ensemble ML methods. The structure comprises two phases: Variant Selection, which identifies optimized versions of classification techniques (e.g., MLP, NB, RBF, KNN, SVM, OneR, and K*, PART, RF, and DT), and Feature Selection, which eliminates irrelevant and redundant features using filter techniques like IG, CFS, and RAE. The framework then applies a variant-based ensemble learning technique, combining the best variants of classification methods to construct an ensemble learning model. The final feature set obtained from the feature selection technique is used to construct models of defect prediction using the ensemble model. The framework is evaluated using three metrics: Accuracy, F-score, and MCC.

There is another study [76] that proposes a novel machine-learning technique for SDP, whose task is to categorize components in a software system that may contain errors. SDP helps to increase the software quality and decreases the cost of testing and development. A major challenge in software defect prediction is that defect datasets are usually unbalanced, meaning that there are more modules without defects than with defects. This makes it difficult for ML models to learn the characteristics of defective modules and classify them correctly. The paper introduces a heterogeneous stacked ensemble-classifier, it is an amalgamation of different types of ML models that are trained on the same data and then combined with another model that learns how to weigh its predictions. The paper claims that this approach handles class imbalance better than existing methods and achieves higher accuracy and AUC scores. The paper evaluates the proposed method on NASA public datasets, which are frequently used in SDP research. This paper compares the proposed method with five basic classifiers (ANN, tree-based classifier, nearest neighbor, SVM and Bayesian classifier) and two modern ensemble techniques (random forest and bagging) A comparison was made. The paper reports that the suggested technique outperforms all other techniques in both accuracy and AUC metrics, and the improvements are statistically significant. This paper concludes that heterogeneous stacking ensemble classifiers are a promising technique for SDP that can deal efficiently with the class imbalance problem. The paper also suggests some future research

directions, such as applying the proposed method to other domains and exploring different ways of stacking base classifiers.

In paper [77], two prominent challenges in software defect prediction are addressed: high dimensionality and data imbalance. The paper introduces a model that combines Recursive Feature Elimination and Partial-Least Squares regression for dimensionality reduction, complemented by the SMOTE (Synthetic Minority Oversampling Technique) to balance data. PLS-R is a statistical technique that reduces the number of attributes by generating new components that capture the maximum variance and correlation with the target variable. RFE, on the other hand, is an attribute selection method that eradicates the least important attributes on the basis of their rank by the classifier. SMOTE is a data augmentation method that creates synthetic samples of the minority class (defective modules) by interpolating between existing samples. The paper further evaluates the performance of various ML algorithms, including DT, LG, SVM, KNNs, XGBoost (a gradient-boosting algorithm using decision trees as base learners), and Stacking Ensemble (a meta-learning technique that combines multiple classifiers using another classifier as a meta-learner). Multiple datasets from the PROMISE.

A study [78] presents a novel approach based on improved self-organizing data mining that discovers hidden patterns and relationships in data without prior knowledge or assumptions. The paper claims that the new method can handle within and across project scenarios, two common settings for software defect prediction. Within project prediction utilizes data from the same project to create and test a prediction model, while cross-project forecasting utilizes data from different projects to build and test a forecasting model. This paper evaluates the new method on 20 publicly available datasets from different software projects and compares it with some existing methods in terms of classification and ranking prediction. Classification prediction assigns each software module a binary label (defective or not) while ranking prediction ranks modules according to their probability of being defective. The paper reports that the new technique gives better performance than existing techniques in classification and ranking prediction, and demonstrates its ability to set up a fundamental relation between bugs and metrics.

A study [79] proposes a framework for CPDP; it is the task of using training data from to calculate software bugs in target projects. In this study, 15 imbalanced learning techniques for cross project defect prediction are studied, including 6 imbalanced ensemble learning (IEL) methods that combine multiple classifiers to treat with imbalance problem. The paper evaluates 15 methods on 20 datasets from 10 projects and compares their performance using different metrics i.e. recall, G-mean, F1-scores, precision, and AUC. The paper also analyzes the impact of different factors such as project size, defect rate, and feature dimension on the performance of the method. The paper finds that for CPDP, IEL methods generally outperform other methods and that the best IEL method is SMOTEBoost, which uses SMOTE (Synthetic Minority Oversampling Technique) to produce new samples for the minority class during boosting.

A paper [80] addresses the problem of multi-source CPDP, a technique aimed at transferring knowledge from source items with labelled data to target items with unlabeled data to predict software defects. In this study, they suggested a new technique based on deep integration for multi-source CPDP, called MTrADL. The technique consists of three steps: (1) attribute extraction, (2) feature integration, and (3) defect prediction. In the first step, the paper uses an Abstract Syntax Tree (AST) and Bi-LSTM (Bidirectional Long-Short Memory) to take out semantic attributes from source modules. In the second step, we use a deep neural network (DNN) to integrate multiple source items and learn high-level attributes that are invariant to the distribution mismatch between different items. In the third step, the paper uses a softmax classifier to forecast the defect label of the target item. The paper evaluates the performance of MTrADL on 19 software items from three public datasets (AEEEM, Relink, and NASA). The paper compares MTrADL with several modern baselines, i.e. TCA+, CDPLS, and DCPDP. The paper reports that MTrADL outperforms all baselines on various evaluation metrics i.e. precision,

accuracy, recall, F-scores, and AUC. The paper also performed an ablation study to analyze the effect of different components of MTrADL. This paper concludes that MTrADL is an efficient and powerful multi-source CPDP method that can provide useful insights for software quality assurance.

There is a study [81] about CVDP (cross-version defect prediction), the task of using data from older versions to forecast defects in newer versions of a software project. It conducts CVDP using the CPDP method, which can use data from different source projects to calculate bugs in test project. It investigates whether inputting multiple older versions of data using a CPDP method is effective for CVDP and compares the output of different CPDP techniques in the CVDP case. It conducts experiments on more than 12 versions of open-source projects, using four CPDP methods: TCA+, TCA, Burak Filter and Naive Bayes. The paper finds that inputting multiple older versions of the data improves some CPDP methods in the CVDP case, but does not work for the best defect prediction method in the paper (TCA+). It also found that using the most recent data published previously is optimal for CVDP and that CPDP data can be useful for CVDP when no CVDP data are available.

A paper [82] proposes a new method called MHCPDP. This technique aims to enhance the accuracy and performance of the prediction models by using multiple source items with different characteristics and applying auto-encoders and multi-source transfer learning techniques. This paper addresses some of the key challenges and limitations of existing HCPDP methods, such as feature heterogeneity, data scarcity, and negative transfer. The paper claims that their method can reduce the feature gap and negative transfer effect between source and target items, and achieve better results than existing HCPDP methods. The paper provides a clear and detailed description of their method, which consists of four steps: feature withdrawal, feature alignment, feature selection, and bugs prediction. The paper also expounds on the theoretical basis and implementation details of each step. The paper conducts extensive experiments on five open-source datasets to validate their method. The paper compares their method to six baseline methods and evaluates it using four metrics: AUC, F-measure, G-mean, and Balance. The paper reports that their method outperforms baseline methods across all metrics and datasets, and demonstrates the effectiveness and robustness of their approach.

In study [83], the focus is on CPDP, the target is to forecast the number of faults in a software project using data from other projects. This study introduces a method that classifies defects into three categories based on defects per thousand lines of code (KLOC). To achieve this classification, the paper utilizes ensemble ML methods, specifically RF and gradient boosting. Ensemble learning integrates many models to increase the accuracy of prediction. The approach's performance is evaluated using open-source projects and compared to baseline techniques like naive Bayes, LR, and DT. The study demonstrates that the given method achieves results comparable to intra-item defect prediction (predicting defects based on a single software item's data) and does better than other CPDP techniques.

In paper [90], the authors tackle the challenge of CPDP and the issue of domain shift, where data distributions between source and target items may significantly differ, leading to suboptimal output. To address this issue, the study introduces a novel CPDP approach based on a Transfer Convolutional Neural Network (TCNN). TCNN is a deep-learning method designed to learn transferable attributes from source items and adapt them to target items. The paper outlines the architecture and training process of TCNN, which includes three key mechanisms: a domain classifier, a feature extractor, and a defect classifier. An attribute extractor learns general attributes from source items, the domain classifier aligns attribute distributions between the training and test projects, and the bug classifier predicts defect labels for target items. The given TCNN method is tested on target datasets and compared to available CPDP methods. The study reports that TCNN achieves superior performance on most datasets compared to existing methods and is effective in handling multiple source items and heterogeneous data.

In paper [91], a new method for SDP is presented, incorporating ML and optimization methods. The method comprises three phases: attribute selection, attribute weighting, and classification. In the first stage, a hybrid model of genetic algorithm and particle swarm optimization is used to select the most relevant features for defect prediction. The feature weighting stage employs a modified PSO to allocate weights to the selected features on the basis of their importance. Finally, the classification phase employs four ML algorithms and uses the weighted features to construct a prediction model. The study evaluates the given technique on software datasets and compares it to existing methods using metrics i.e. accuracy, F-measure, and AUC. The output demonstrate that the suggested technique outperforms existing methods on all metrics. Additionally, the paper conducts analysis to validate the implication of the output results.

5 Conclusion

CPDP is a capable technique to enhance software quality and minimize maintenance costs by utilizing data from source projects to forecast defects in target projects. This comprehensive review delves into a multitude of research endeavors and strategies within the realm of CPDP, specifically focusing on machine learning techniques. The goal is to offer valuable insights and guidance for both researchers and practitioners interested in this domain. CPDP serves as an invaluable solution when defect data is limited or nonexistent in target projects. By leveraging data from other projects, CPDP enables the prediction of software defects in target projects, ultimately elevating the efficacy of software quality assurance processes. This evolving field continues to contribute to the advancement of software engineering practices, thereby optimizing software development and maintenance efforts.

1- **Challenges and Open Issues:** CPDP using Machine Learning faces several challenges and open issues, such as data selection, feature extraction from heterogeneous sources, model selection, evaluation metrics, and transfer learning. These challenges must be addressed to increase the accuracy and generality of CPDP models.

2- **Feature Selection and Model Selection:** CPDP research explores various feature selection methods and ML models. Ensemble methods, genetic algorithms, self-attention mechanisms, and deep learning have proven effective in improving defect prediction accuracy.

3- **Evaluation Indicators and Performance:** CPDP model evaluation involves multiple indicators, including precision rate, accuracy rate, recall rate, G-mean, F1-score, and AUC-ROC. The choice of evaluation metric is crucial for accurately assessing model performance.

4- **Performance and Robustness:** Studies evaluating CPDP models on real datasets from repositories like AEEEM and PROMISE demonstrate that ML-based CPDP methods typically outperform traditional statistical approaches, indicating their potential for performance improvements.

5- **Limitations and Future Directions:** Despite progress, CPDP using ML still faces limitations related to data quality, class imbalance, and feature selection. Future research should focus on addressing these challenges, exploring new features, and developing interpretable models.

6- **Recommendations:** Based on the review, it is recommended that researchers and practitioners consider hybrid approaches that combine statistical and machine learning techniques. Utilizing ensemble learning, addressing class imbalance, and employing feature selection techniques can increase the accuracy and precision of CPDP models. These recommendations aim to advance the field and improve software defect prediction practices.

In conclusion, this review paper provides a complete summary of modern methods and techniques for CPDP using machine learning. The findings show that the machine learning-based CPDP method shows great promise in improving software quality and predicting target project defects. As the field is growing continuously, further research and development are required to address existing challenges and open questions, ultimately contributing to stronger and more efficient CPDP models. This review can provide a valuable reference for researchers to enhance the field of CPDP and make informed decisions while employing ML techniques for cross-project defect prediction.

Reference

1. Khleel, N. A. A., & Nehéz, K. (2023). A novel approach for software defect prediction using CNN and GRU based on the SMOTE Tomek method. Journal of Intelligent Information Systems, 1-35.
2. Esteves, G., Figueiredo, E., Veloso, A., Viggiato, M., & Ziviani, N. (2020). Understanding machine learning software defect predictions. Automated Software Engineering, 27(3-4), 369-392.
3. Agrawal, A., & Malhotra, R. (2022). Cross project defect prediction for open source software. International Journal of Information Technology, 14(1), 587-601.
4. Jahanshahi, H., Cevik, M., & Başar, A. (2021). Moving from cross-project defect prediction to heterogeneous defect prediction: a partial replication study. arXiv preprint arXiv:2103.03490.
5. Rahman, A. U., Abbas, S., Gollapalli, M., Ahmed, R., Aftab, S., Ahmad, M. ... & Mosavi, A. (2022). Rainfall prediction system using machine learning fusion for smart cities. Sensors, 22(9), 3504.
6. Ahmed, U., Issa, G. F., Khan, M. A., Aftab, S., Khan, M. F., Said, R. A., & Ahmad, M. (2022). Prediction of diabetes empowered with fused machine learning. IEEE Access, 10, 8529-8538.
7. Aziz, N., & Aftab, S. (2021). Data mining framework for nutrition ranking: Methodology: SPSS modeller. International Journal of Technology, Innovation and Management (IJTIM), 1(1), 85-95.
8. Daoud, M. S., Fatima, A., Khan, W. A., Khan, M. A., Abbas, S., Ihnaini, B., & Aftab, S. (2021). Joint Channel and Multi-User Detection Empowered with Machine Learning.
9. Aftab, S., Alanazi, S., Ahmad, M., Khan, M. A., Fatima, A., & Elmitwally, N. S. (2021). Cloud-Based Diabetes Decision Support System Using Machine Learning Fusion. Computers, Materials & Continua, 68(1).
10. Ahmad, M., Alfayad, M., Aftab, S., Khan, M. A., Fatima, A., Shoaib, B., & Elmitwal, N. S. (2021). Data and Machine Learning Fusion Architecture for Cardiovascular Disease Prediction. Computers, Materials & Continua, 69(2).
11. Shabib Aftab, M. A., Hameed, N., Bashir, M. S., Ali, I., & Nawaz, Z. (2018). Rainfall prediction in Lahore City using data mining techniques. International journal of advanced computer science and applications, 9(4).
12. Iqbal, A., & Aftab, S. (2019). A feed-forward and pattern recognition ANN model for network intrusion detection. International Journal of Computer Network and Information Security, 11(4), 19.
13. Ahmad, M., Aftab, S., Bashir, M. S., Hameed, N., Ali, I., & Nawaz, Z. (2018). SVM optimization for sentiment analysis. International Journal of Advanced Computer Science and Applications, 9(4).
14. Khan, M. A., Elmitwally, N. S., Abbas, S., Aftab, S., Ahmad, M., Fayaz, M., & Khan, F. (2022). Software defect prediction using artificial neural networks: A systematic literature review. Scientific Programming, 2022.
15. Matloob, F., Ghazal, T. M., Taleb, N., Aftab, S., Ahmad, M., Khan, M. A., & Soomro, T. R. (2021). Software defect prediction using ensemble learning: A systematic literature review. IEEE Access, 9, 98754-98771.
16. Matloob, F., Aftab, S., Ahmad, M., Khan, M. A., Fatima, A., Iqbal, M., & Elmitwally, N. S. (2021). Software defect prediction using supervised machine learning techniques: A systematic literature review. Intelligent Automation & Soft Computing, 29(2), 403-421.

17. Ahmad, M., Aftab, S., Bashir, M. S., & Hameed, N. (2018). Sentiment analysis using SVM: a systematic literature review. International Journal of Advanced Computer Science and Applications, 9(2).
18. Aftab, S., Ahmad, M., Hameed, N., Bashir, M. S., Ali, I., & Nawaz, Z. (2018). Rainfall prediction using data mining techniques: A systematic literature review. International journal of advanced computer science and applications, 9(5).
19. Iqbal, A., Aftab, S., & Matloob, F. (2019). Performance analysis of resampling techniques on class imbalance issue in software defect prediction. Int. J. Inf. Technol. Comput. Sci, 11(11), 44-53.
20. Rahman, M. H., Sharmin, S., Islam, M. S., Khaled, S. M., & Sarwar, S. M. An Attribute Selection Process for Cross-Project Software Defect Prediction.
21. Huda, S., Liu, K., Abdelrazek, M., Ibrahim, A., Alyahya, S., Al-Dossari, H., & Ahmad, S. (2018). An ensemble oversampling model for class imbalance problem in software defect prediction. IEEE access, 6, 24184- 24195.Qiao, L., Li, X., Umer, Q., & Guo, P. (2020). Deep learning based software defect prediction. Neurocomputing, 385, 100-110.
22. Qiao, L., Li, X., Umer, Q., & Guo, P. (2020). Deep learning based software defect prediction. Neurocomputing, 385, 100-110.
23. Iqbal, A., & Aftab, S. (2020). Prediction of defect prone software modules using MLP based ensemble techniques. International Journal of Information Technology and Computer Science, 12(3), 26-31.
24. Lei, T., Xue, J., Wang, Y., Niu, Z., Shi, Z., & Zhang, Y. (2022). WCM-WTrA: A Cross-Project Defect Prediction Method Based on Feature Selection and Distance-Weight Transfer Learning. Chinese Journal of Electronics, 31(2), 354-366.
25. Bala, Y. Z., Samat, P. A., Sharif, K. Y., & Manshor, N. (2022). Improving Cross-Project Software Defect Prediction Method through Transformation and Feature Selection Approach. IEEE Access, 11, 2318-2326.
26. Iqbal, A., Aftab, S., Ullah, I., Bashir, M. S., & Saeed, M. A. (2019). A feature selection based ensemble classification framework for software defect prediction. International Journal of Modern Education and Computer Science, 11(9), 54.
27. Xu, Z., Zhang, T., Keung, J., Yan, M., Luo, X., Zhang, X., & Tang, Y. (2021). Feature selection and embedding based cross project framework for identifying crashing fault residence. Information and Software Technology, 131, 106452.
28. Dada, E. G., Oyewola, D. O., Joseph, S. B., & Duada, A. B. (2021). Ensemble machine learning model for software defect prediction. Adv. Mach. Learn. Artif. Intell, 2, 11-21.
29. Matloob, F., Aftab, S., & Iqbal, A. (2019). A Framework for Software Defect Prediction Using Feature Selection and Ensemble Learning Techniques. International Journal of Modern Education & Computer Science, 11(12).
30. Pal, S. (2021). Generative Adversarial Network-based Cross-Project Fault Prediction. arXiv preprint arXiv:2105.07207.
31. Li, Y., Wen, M., Liu, Z., & Zhang, H. (2022). Using Cost-cognitive Bagging Ensemble to Improve Crossproject Defects Prediction. Journal of Internet Technology, 23(4), 779-789.
32. Iqbal, A., & Aftab, S. (2020). A Classification Framework for Software Defect Prediction Using Multi-filter Feature Selection Technique and MLP. International Journal of Modern Education & Computer Science, 12(1).
33. Huang, Q., Ma, L., Jiang, S., Wu, G., Song, H., Jiang, L., & Zheng, C. (2022). A cross-project defect prediction method based on multi-adaptation and nuclear norm. IET Software, 16(2), 200-213.
34. Aftab, S., Abbas, S., Ghazal, T. M., Ahmad, M., Hamadi, H. A., Yeun, C. Y., & Khan, M. A. (2023). A CloudBased Software Defect Prediction System Using Data and Decision-Level Machine Learning Fusion. Mathematics, 11(3), 632.
35. Xu, Z., Liu, J., Luo, X., Yang, Z., Zhang, Y., Yuan, P., & Zhang, T. (2019). Software defect prediction based on kernel PCA and weighted extreme learning machine. Information and Software Technology, 106, 182-200.

36. Tang, S., Huang, S., Zheng, C., Liu, E., Zong, C., & Ding, Y. (2021). A novel cross-project software defect prediction algorithm based on transfer learning. Tsinghua Science and Technology, 27(1), 41-57.
37. Iqbal, A., Aftab, S., Ali, U., Nawaz, Z., Sana, L., Ahmad, M., & Husen, A. (2019). Performance analysis of machine learning techniques on software defect prediction using NASA datasets. International Journal of Advanced Computer Science and Applications, 10(5).
38. Sinaga, B. L., Ahmad, S., Abas, Z. A., & Jalil, I. E. A. (2022). A recommendation system of training data selection method for cross-project defect prediction. Indonesian Journal of Electrical Engineering and Computer Science, 27(2), 990-1006.
39. Kondo, M., Bezemer, C. P., Kamei, Y., Hassan, A. E., & Mizuno, O. (2019). The impact of feature reduction techniques on defect prediction models. Empirical Software Engineering, 24, 1925-1963.
40. Abbas, S., Aftab, S., Khan, M. A., Ghazal, T. M., Hamadi, H. A., & Yeun, C. Y. (2023). Data and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System. Computers, Materials & Continua, 75(3).
41. Daoud, M. S., Aftab, S., Ahmad, M., Khan, M. A., Iqbal, A., Abbas, S., & Ihnaini, B. (2022). Machine learning empowered software defect prediction system.
42. Noreen, S., Faiz, R. B., Alyahya, S., & Maddeh, M. (2022). Performance Evaluation of Convolutional Neural Network for Multi-Class in Cross Project Defect Prediction. Applied Sciences, 12(23), 12269.
43. Yamamoto, H., Wang, D., Rajbahadur, G. K., Kondo, M., Kamei, Y., & Ubayashi, N. (2023, March). Towards Privacy Preserving Cross Project Defect Prediction with Federated Learning. In 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 485-496). IEEE.
44. Zhou, T., Sun, X., Xia, X., Li, B., & Chen, X. (2019). Improving defect prediction with deep forest. Information and Software Technology, 114, 204-216.
45. Shippey, T., Bowes, D., & Hall, T. (2019). Automatically identifying code features for software defect prediction: Using AST N-grams. Information and Software Technology, 106, 142-160.
46. Zhao, Y., Zhu, Y., Yu, Q., & Chen, X. (2021). Cross-Project defect prediction method based on manifold feature transformation. Future Internet, 13(8), 216.
47. Bennin, K. E., Tahir, A., MacDonell, S. G., & Börstler, J. (2022). An empirical study on the effectiveness of data resampling approaches for cross-project software defect prediction. IET Software, 16(2), 185-199.
48. Zou, Q., Lu, L., Qiu, S., Gu, X., & Cai, Z. (2021). Correlation feature and instance weights transfer learning for cross project software defect prediction. IET Software, 15(1), 55-74.
49. Zhong, Y., Song, K., Lv, S., & He, P. (2021). An Empirical Study of Software Metrics Diversity for CrossProject Defect Prediction. Mathematical Problems in Engineering, 2021, 1-11.
50. Li, Z., Jing, X. Y., & Zhu, X. (2018). Progress on approaches to software defect prediction. Iet Software, 12(3), 161-175.
51. Yu, Q., Qian, J., Jiang, S., Wu, Z., & Zhang, G. (2019). An empirical study on the effectiveness of feature selection for cross-project defect prediction. IEEE Access, 7, 35710-35718.
52. Qu, Y., Chen, X., Zhao, Y., & Ju, X. (2018). Impact of hyper parameter optimization for cross-project software defect prediction. International Journal of Performability Engineering, 14(6), 1291.
53. Liu, W., Wang, B., & Wang, W. (2021). Deep learning software defect prediction methods for cloud environments research. Scientific Programming, 2021, 1-11.
54. Wang, K., Liu, L., Yuan, C., & Wang, Z. (2021). Software defect prediction model based on LASSO– SVM. Neural Computing and Applications, 33, 8249-8259.
55. Saifudin, A., & Heryadi, Y. (2019, November). Ensemble Undersampling to Handle Unbalanced Class on Cross-Project Defect Prediction. In IOP Conference Series: Materials Science and Engineering (Vol. 662, No. 6, p. 062012). IOP Publishing.
56. Ulumi, D. I., & Siahaan, D. (2019, July). Weighted knn using grey relational analysis for cross-project defect prediction. In Journal of Physics: Conference Series (Vol. 1230, No. 1, p. 012062). IOP Publishing.

57. Malhotra, R., & Kamal, S. (2019). An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. Neurocomputing, 343, 120-140.
58. Fan, G., Diao, X., Yu, H., Yang, K., & Chen, L. (2019). Software defect prediction via attention-based recurrent neural network. Scientific Programming, 2019.
59. Jacob, R. J., Kamat, R. J., Sahithya, N. M., John, S. S., & Shankar, S. P. (2021, October). Voting based ensemble classification for software defect prediction. In 2021 IEEE Mysore Sub Section International Conference (MysuruCon) (pp. 358-365). IEEE.
60. Ali, A. R., Rehman, A. U., Nawaz, A., Ali, T. M., & Abbas, M. (2022, May). An Ensemble Model for Software Defect Prediction. In 2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2) (pp. 1-5). IEEE.
61. Manjula, C., & Florence, L. (2018). SOFTWARE DEFECT PREDICTION USING DEEP BELIEF NETWORK WITH L1-REGULARIZATION BASED OPTIMIZATION. International Journal of Advanced Research in Computer Science, 9(1).
62. Cetiner, M., & Sahingoz, O. K. (2020, July). A comparative analysis for machine learning based software defect prediction systems. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.
63. Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2023). Software Defect Prediction Analysis Using Machine Learning Techniques. Sustainability, 15(6), 5517.
64. Saheed, Y. K., Longe, O., Baba, U. A., Rakshit, S., & Vajjhala, N. R. (2021). An ensemble learning approach for software defect prediction in developing quality software product. In Advances in Computing and Data Sciences: 5th International Conference, ICACDS 2021, Nashik, India, April 23–24, 2021, Revised Selected Papers, Part I 5 (pp. 317-326). Springer International Publishing.
65. Zong, X., Li, G., Zheng, S., Zou, H., Yu, H., & GAO, S. (2023). Heterogeneous cross-project defect prediction via optimal transport. IEEE Access, 11, 12015-12030.
66. Hu, Z., & Zhu, Y. (2023). Cross-project defect prediction method based on genetic algorithm feature selection. Engineering Reports, e12670.
67. Bhat, N. A., & Farooq, S. U. (2023). An empirical evaluation of defect prediction approaches in within-project and cross-project context. Software Quality Journal, 1-30.
68. Aljaidi, M., Gul, S., Faiz, R., Samara, G., Alsarhan, A., & al-Qerem, A. (2023). Impact Evaluation of Significant Feature Set in Cross Project for Defect Prediction through Hybrid Feature Selection in Multiclass. bioRxiv, 2023-07.
69. Nevendra, M., & Singh, P. (2022). Cross-Project Defect Prediction with Metrics Selection and Balancing Approach. Applied Computer Systems, 27(2), 137-148.
70. Govinda, N. N., Lohith, R., Jha, R. K., & Gururaj, H. L. (2023). Cross-Project Fault Prediction using Artificial Intelligence. International Journal of Bioinformatics and Intelligent Computing, 2(1), 73-81.
71. Vashisht, R., & Rizvi, S. A. M. (2023). Addressing Noise and Class Imbalance Problems in Heterogeneous Cross-Project Defect Prediction: An Empirical Study. International Journal of e-Collaboration (IJeC), 19(1), 1- 27.
72. Faiz, R. B., Shaheen, S., Sharaf, M., & Rauf, H. T. (2023). Optimal Feature Selection through Search-Based Optimizer in Cross Project. Electronics, 12(3), 514.
73. Sasankar, M. P., & Sakarkar, G. (2022). Cross Project Defect Prediction using Deep Learning Techniques. In International Conference on Artificial Intelligence & Big Data Analytics.
74. Wen, W., Zhang, R., Wang, C., Shen, C., Yu, M., Zhang, S., & GAO, X. (2022). A Cross-Project Defect Prediction Model Based on Deep Learning with Self-Attention. IEEE Access, 10, 110385-110401.
75. Ali, U., Aftab, S., Iqbal, A., Nawaz, Z., Bashir, M. S., & Saeed, M. A. (2020). Software defect prediction using variant based ensemble learning and feature selection techniques. International Journal of Modern Education & Computer Science, 12(5).
76. Goyal, S. (2020, November). Heterogeneous stacked ensemble classifier for software defect prediction. In 2020 sixth international conference on parallel, distributed and grid computing (PDGC) (pp. 126-130). IEEE.

77. Mehta, S., & Patnaik, K. S. (2021). Improved prediction of software defects using ensemble machine learning techniques. Neural Computing and Applications, 33, 10551-10562.
78. Zhang, Q., & Ren, J. (2022). Software-defect prediction within and across projects based on improved selforganizing data mining. The Journal of Supercomputing, 78(5), 6147-6173.
79. Qiu, S., Lu, L., Jiang, S., & Guo, Y. (2019). An investigation of imbalanced ensemble learning methods for cross-project defect prediction. International Journal of Pattern Recognition and Artificial Intelligence, 33(12), 1959037.
80. Zhang, J., Wang, W., He, Y., & Li, X. (2021, March). Multi-source cross-project software defect prediction based on deep integration. In Journal of Physics: Conference Series (Vol. 1861, No. 1, p. 012075). IOP Publishing.
81. Amasaki, S. (2018, October). Cross-version defect prediction using cross-project defect prediction approaches: Does it work? In Proceedings of the 14th International Conference on predictive models and data analytics in software engineering (pp. 32-41).
82. Wu, J., Wu, Y., Niu, N., & Zhou, M. (2021). MHCPDP: Multi-source heterogeneous cross-project defect prediction via multi-source transfer learning and autoencoder. Software Quality Journal, 29(2), 405-430.
83. Goel, L., Sharma, M., Khatri, S. K., & Damodaran, D. (2020). Prediction of cross project defects using ensemble based multinomial classifier. EAI Endorsed Transactions on Scalable Information Systems, 7(25), e5- e5.
84. Ahmed, F., Asif, M. and Saleem, M., 2023. Identification and Prediction of Brain Tumor Using VGG-16 Empowered with Explainable Artificial Intelligence. International Journal of Computational and Innovative Sciences, 2(2), pp.24-33.
85. Saleem, M., Khan, M.S., Issa, G.F., Khadim, A., Asif, M., Akram, A.S. and Nair, H.K., 2023, March. Smart Spaces: Occupancy Detection using Adaptive Back-Propagation Neural Network. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-6). IEEE.
86. Athar, A., Asif, R.N., Saleem, M., Munir, S., Al Nasar, M.R. and Momani, A.M., 2023, March. Improving Pneumonia Detection in chest X-rays using Transfer Learning Approach (AlexNet) and Adversarial Training. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-7). IEEE. [1]
87. Sajjad, G., Khan, M.B.S., Ghazal, T.M., Saleem, M., Khan, M.F. and Wannous, M., 2023, March. An Early Diagnosis of Brain Tumor Using Fused Transfer Learning. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-5). IEEE.
88. Saleem, M., Abbas, S., Ghazal, T.M., Khan, M.A., Sahawneh, N. and Ahmad, M., 2022. Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques. Egyptian Informatics Journal, 23(3), pp.417-426.
89. Saleem, M., Khadim, A., Fatima, M., Khan, M.A., Nair, H.K. and Asif, M., 2022, October. ASSMA-SLM: Autonomous System for Smart Motor-Vehicles integrating Artificial and Soft Learning Mechanisms. In 2022 International Conference on Cyber Resilience (ICCR) (pp. 1-6). IEEE.
90. Qiu, S., Xu, H., Deng, J., Jiang, S., & Lu, L. (2019). Transfer convolutional neural network for cross-project defect prediction. Applied Sciences, 9(13), 2660.
91. Yao, Z., Sun, L., Zhang, T., & Wang, J. (2019, June). The cross-project defect prediction based on PSO and Feature Dependent Naive Bayes. In Journal of Physics: Conference Series (Vol. 1237, No. 2, p. 022126). IOP Publishing
92. Abualkishik, A., Saleem, M., Farooq, U., Asif, M., Hassan, M. and Malik, J.A., 2023, March. Genetic Algorithm Based Adaptive FSO Communication Link. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-4). IEEE.
93. Malik, J.A. and Saleem, M., 2022. Blockchain and Cyber-Physical System for Security Engineering in the Smart Industry. In Security Engineering for Embedded and Cyber-Physical Systems (pp. 51-70). CRC press.