International Journal of Computations, Information and Manufacturing (IJCIM) 3(2) -2023

Contents available at the publisher website: G A F T I M . C O M

International Journal of Computations, Information and Manufacturing (IJCIM)

Journal homepage: https://journals.gaftim.com/index.php/ijcim/index

Software Defect Prediction Using Clustering: A Comprehensive Literature Review

Amna Batool

Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan

A R T I C L E I N F O

Keywords:
Software Defect Prediction, Clustering, Software Engineering

Received: Sep, 21, 2023
Accepted: Oct, 29, 2023
Published: Dec, 22, 2023

A B S T R A C T

Anticipating software defects prior to the testing phase proves advantageous for efficient resource allocation to develop the high-quality software, a necessity for any organization. Machine learning (ML) methodologies play a pivotal role in addressing these issues, leading to the creation of numerous predictive models designed to categorize software modules as either defective or non-defective. Several obstacles hinder the analysis of software data that is defected, encompassing issues like redundancy, correlation, irrelevant features, missing data points, and an unbalance distribution between faulty and non-faulty classes. Both supervised and unsupervised machine learning techniques have garnered global attention from practitioners and researchers as viable approaches to tackle these challenges, yielding noticeable enhancements in defect prediction accuracy. This review paper examines clustering unsupervised machine learning technique developed for software defect prediction spanning the years 2017 to 2023 and covered the 15 researches.

1. INTRODUCTION

Software engineering is a branch of engineering that produces a wide range of software products that are reliable, effective, and efficient at their jobs. It covers every facet of software development, from the specification of the program to user maintenance once it is delivered to them. This suggests that the procedure used to produce software and the effectiveness of its testing are the only factors that affect the software's quality (Mafarja, M, et al., 2023). In other words the goal of software engineering has traditionally been to produce high-quality software with little resources. Regrettably, errors have the potential to manifest at any stage of the software development process and the utilization of software has become intertwined with daily work routines and business operations. The emergence of defects within software can potentially trigger significant economic disruptions. Hence, one of the paramount objectives within the software industry is the ability to anticipate software defects beforehand. This anticipation aids in the identification of classes and modules that are prone to errors, subsequently necessitating modifications or the application of diverse testing strategies. A crucial part of the development process is the testing phase. Improving software quality and cutting overall costs are the two main goals of the testing process (Thirumoorthy, K., & Britto, J. J. J. 2022). The pursuit of constructing high-quality software places considerable emphasis on defect prediction, as considerable time and effort are expended on testing and debugging. The quantity of flaws in software determines its quality. A high defect count results in decreased customer satisfaction, more organizational costs and resource consumption, and delayed testing. Defect prediction methodologies are introduced with the intention of streamlining software testing and debugging endeavors. By highlighting software

components with a higher likelihood of being faulty, these methodologies offer guidance to developers, assisting them in effectively prioritizing their efforts. Undetected defects or vulnerabilities can lead to escalating expenses, primarily in the form of costs associated with fixing or remediation. Employing automated tools for static analysis offers an immediate understanding of the root causes of these issues, simplifying the resolution of longstanding problems. One of the software engineering research topics that is still being extensively studied is software defect prediction (SDP) (Khalid, A., Badshah, G, et al., 2023). The testing procedure is optimally facilitated by the use of Software Fault Prediction (SFP/ SDP). Software modules are categorized as either defect-prone or non-defect-prone by the defect prediction process. Software effect area exploration commonly makes use of defect prediction techniques such clustering (Aftab, S., Alanazi, S, et al., 2023) , statistical approaches, mixed algorithms, neural network-based metrics, black box testing, white box testing, and machine learning. In the past, conventional methods for defect detection included activities such as reviews, walkthroughs, code inspections, and testing, but defects could also be stumbled upon by chance. If the erroneous software modules are found before the testing stage, the cost of testing can be greatly reduced and also the entire cost of development can be decreased, with the assurance of excellent quality if only those software modules that are identified as defective are shortlisted for testing, as testing operations demand a large amount of resources (Bowes, D., Hall, T., & Petrić, J. 2018). It results in early fixes and, ultimately, on-time delivery of maintainable software, satisfying the client and fostering his trust in the development team. Focusing on the software modules where problems were most likely to emerge would save a large amount of time if the development team knew in advance which software modules were problematic. Additionally, techniques rooted in Artificial Intelligence, such as Machine Learning (Supervised and Unsupervised ML), are also employed for this context (Ayon, S. I, 2019).

The primary aim of this research is to conduct a thorough examination of the utilization of Clustering, an unsupervised Machine Learning technique, in the domain of SDP (Gong, L., Jiang, S., & Jiang, L. 2019). We have reviewed publications up to the year 2023, taking into account various dimensions, such as the choice of a specific machine learning algorithm, the unique contributions made by authors, and the identification of research gaps that could potentially shape the future directions of this field.

1.1. Machine Learning Techniques

In recent years, machine learning techniques have garnered widespread recognition owing to their impressive capabilities and technical advancements. In recent fiercely competitive business landscape, machine learning (ML) plays a vital role in expediting digital transformation and in the era of automation. Machine learning techniques are applied to software to achieve quality, maintainability, and reusability by identifying defects, faults, ambiguities, and unpleasant smells. ML also helps in researching many areas in software engineering (Usman-Hamza, et al., 2019). These techniques, grounded in statistical methods, empower algorithms to make predictions and categorizations, unveiling valuable insights in the domain of software. These insights significantly influence decision-making processes and key performance indicators associated with software applications. Many researchers have focused on machine learning techniques to solve the binary classification problems, including Network Intrusion Detection, Sentiment Analysis (Yang, Y., Yang, J., & Qian, H, 2018), lexicon driven sentiment analysis, Latest transformations in scrum, Rainfall Prediction, cross project defect prediction and SDP. In the context of research on SDP, two primary machine learning techniques, Supervised and Unsupervised are prominent. Additionally, Semi-supervised and Reinforcement Learning techniques also exist but find limited application in SDP. Notably, Deep Learning techniques and Ensemble Learning are also gaining importance in SDP. Using the ensemble learning technique, many models can be combined into a single group model in machine learning. Voting, bagging, boosting, and stacking are examples of the widely used ensemble approach.

1.2. Supervised Machine Learning

Supervised Learning emerges as the most widely adopted method for predicting software bugs. Its core principle revolves around using datasets

having labels to train the algorithms for dependable class prediction. Two classification approaches exist within supervised learning: one involves dividing data into training, testing, and validation sets, while the other employs n cross-validation techniques (Balogun, A., et al.,2019).

Key characteristics of supervised machine learning include:

The primary objective of the model is to determine labels / values for unseen instances based on their characteristics. Labels denote the values associated with a data instance's chosen target attribute, predicted by utilizing information derived from other attributes.

Classification problems arise when labels represent categories, whereas regression problems involve labels as continuous numerical values.

To forecast the likelihood of software defects, Incorporating each software instance as a data point, with software attributes defining its characteristics, and determining whether the software exhibits defects or not serves as a fundamental aspect of our methodology. This approach assists in the identification of problematic modules. After gathering flawed data, it is possible to create an ML model to assess new software for faults early in the development process, ensuring software reliability and quality. To address class imbalance issues, common data sampling techniques are employed, as software defects are often a minority class compared to non-defective instances, potentially affecting prediction model accuracy.

There are many supervised machine learning methods like Support Vector Machine (SVM), Decision Tree(DT), Random Forest, Naïve Bayes(NB), Linear Regression, Logistic Regression, K-Nearest Neighbour, Artificial Neural Network, Multi Layer Preceptron.

1.3. Unsupervised Learning:

Unsupervised learning (Tang, S., Huang, et al., 2022) is a machine-based technique that differs from supervised learning in that it doesn't rely on labeled training data but instead utilizes unlabeled datasets. In this approach, it identifies concealed patterns within the provided data without human guidance. Unlike supervised learning, unsupervised learning is not directly applicable to regression or classification problems since there are no corresponding output data corresponds to entered data.

Unsupervised learning primarily relies on the method of clustering (Ayon, S. I, 2019), a valuable tool for classifying data objects into separate groups or clusters. Clustering is instrumental in structuring a set of objects by grouping those that exhibit similarity. In the realm of software quality analysis, clustering is applied to datasets that comprise measurements from both defective and non-defective software. Within this framework, the data is divided into two clusters based on the presence of defects. Following this division, the datasets undergo suitable clustering techniques to predict faulty and faultless modules using software defect prediction algorithms. Moreover, clustering methods can also serve as a means to evaluate the quality of software.

A notable feature of clustering is the capability to function effectively in the presence of noise, without necessitating the predefinition of clusters. There are many clustering techniques used for SDP like K-means Clustering (Matloob, F, et al.,2021) Hierarchical Clustering, Density-based Clustering, Neural Network Clustering, Expectation Maximization Clustering, Spectral Clustering and Optimization Clustering.

Various research studies have demonstrated that Clustering Software Defect Prediction models exhibit comparable predictive performance to Supervised Defect Prediction models. This suggests that clustering models for defect prediction may not be as challenging as previously believed and should be considered when there is limitation on labeled training data. Furthermore, scholars have noted that the attributes of a dataset substantially influence predictive accuracy, whether employing unsupervised or supervised models for defect detection in software. Therefore, a more thorough exploration of these dataset properties' relevance is warranted. Furthermore, investigating the interaction between the learning system and the dataset could prove beneficial, as it remains a question of which specific learning approach, whether supervised or unsupervised, excels under different circumstances.

2. RELATED WORK

This section presents an insight in recent investigations in software defect detection conducted by a variety of researchers. The

examination centers on research that has formulated a faultless prediction model for anticipating software glitches through the utilization of clustering, an unsupervised machine learning technique.

In the research (Ni, C., Liu, W., Gu, Q., Chen, X., & Chen, D, 2017), they introduced a novel clustering technique designed to group the modules. They presented the Hybrid Elitist Self-Adaptive Multi-Population Social Mimic Optimization technique (ESAMP-SMO). Key attributes of ESAMP-SMO algorithm include self-adaptation, multi-population management, elitism, crossover, and mutation. The research methodology began with an exploration of the maximum number of generations, followed by the generation of initial candidate solutions and the division of the population into sub-populations. Subsequently, a social mimic optimization process was applied to each sub-population, culminating in the merging of all sub-populations and the identification of a new global leader solution. After a careful examination, the cross and mutation operations were applied, leading to the final output solution. The evaluation of approach involved the use of various performance metrics, including Precision (P), accuracy (ACC), recall (R), sensitivity, F1-score, specificity, false negative rate (FNR), Fowlkes-Mallows Index (FMI), false positive rate (FPR) and Youden's index (YI). These metrics were applied to three well-known NASA benchmark datasets, namely, CM1, KC1, and JM1. In order to gauge the effectiveness, they conducted a benchmark comparison with evolutionary-based clustering methods such as Crow Search Algo, Particle Swarm Optimization, Jaya Optimization Algorithm, Gray Wolf Optimization and Genetic Algorithm. The results of the performance comparison analysis unambiguously demonstrate the superior performance of our proposed clustering technique when compared to the other competing approaches. The research aimed to enhance the model's performance concerning dataset accuracy and precision when compared to prior studies. To achieve this goal, they employed K-means clustering to categorize class labels and subsequently applied classification models to specific features. Their analysis involved three ML classification models: Linear SVC (SVM), Gaussian NB (NB), and Random Forest (RF) with Stacking ensemble methodology, where NB served as the base model, RF and SVM as member models. Particle Swarm Optimization(PSO) was used to optimize the ML models. They assessed model performance using accuracy, precision, F-measure, recall, confusion matrix and performance error metrics. The results showed that all optimized ML and ML models achieved the maximum potential, with SVM and optimized SVM models outperforming the rest with the highest accuracy rates of 99% and 99.80%, respectively. The accuracy scores for RF, Optimized RF, NB, Optimized NB, and ensemble approaches were 98.70%, 99.50%,93.90%, 93.80%, 98.80%, and 97.60%, respectively.

Many defect-related features have been developed for SDP by researchers in order to increase prediction performance. However, the performance of defect prediction will be significantly diminished by feature redundancy (FR) and irrelevance brought on by the growing dimensions of data. Researchers have suggested a number of data dimensionality reduction techniques to address the issues. The two types of approaches that make up these techniques are feature selection and feature extraction. The two types of approaches both have advantages and disadvantages, though. In order to enhance the performance of SDP, they suggested a Hybrid Feature Dimensionality Reduction Approach (HFDRA) for SDP in study (Usman-Hamza, et al., 2019). This approach combines the two various types of approaches. The two stages of the HFDRA technique are feature selection and feature extraction. In the feature selection stage, HFDRA first uses a clustering method to separate the original features into various feature subsets. Then, in the feature extraction stage, each feature subset's dimensionality is decreased using kernel principal component analysis (KPCA). The prediction model is then created using the reduced-dimensional data. They defined the correlation among features (FF-correlation) and the correlation between features and defect types (FC-correlation). 22 projects from AEEEM, SOFTLAB, MORP, and ReLink are used as experiment objects in the empirical investigation. In this study, they first compared their method with three state-of-the-art approaches and seven baseline methods named as ALL, IG, MIC, KPCA, CFS, FCBF and ReliefF. NB and SVM are two separate classifiers that they used in this paper's

experiment to develop a defect prediction model. Next, they examined the connection between FR and forecast accuracy. AUC, F1, and MCC three measures are also used to assess the success of four different prediction algorithms. On each project, the experimental outcome is also based on the 10-fold cross-validation average value. The results of the experiments demonstrate that their strategy outperforms cutting-edge data dimensionality reduction techniques for fault prediction.

The majority of CPDP approaches base their defect prediction models on how closely two projects' feature spaces or data are related. The problem arises in case when the target project has a tiny amount of label data. Therefore, it is challenging to get acceptable prediction performance using these methods when the distribution between the source project and the target project is substantially different. This paper (Huda, S., Liu, K., Abdelrazek, et al., 2018) developed a CPDP approach based on semisupervised clustering, known as Tsbagging, to address this issue. The clustering stage and the ensemble stage make up the majority of tsbagging. The data from the source project were clustered using a semisupervised method (TSCluster), and various weights were then applied to the data in accordance with the clustering results. They employed bagging to train a number of classifiers and integrate them to forecast the defect in the ensemble stage, which was dependent on the weights provided in the clustering step. In this study, we analyze the effectiveness of defect prediction using the F1-measure, MCC, g-measure, and balance. The experimental results demonstrate that Tsbagging outperforms other SDP approaches in terms of performance. The primary goal of the study conducted in reference (Usman-Hamza, et al., 2019) was to evaluate the efficiency of integrating clustering techniques with feature selection methods as a strategy to address the software defect prediction challenge. The study examined seven preprocessed datasets from the National Aeronautics and Space Administration (NASA) utilizing a variety of clustering techniques, including K-MEANS, Density-Based (DB), Expectation Maximization (EM), Self-Organizing Maps (SOM), Farthest First, Neural Networks, Learning Vector Quantization (LVQ), and X-Means. To evaluate and compare a variety of feature selection techniques, the researchers conducted an analysis of software defect prediction using PSO, Cuckoo, Bat, and Grey Wolf Optimizer (GWO). Their framework comprises three key stages: Feature Selection, Clustering & Decision Making. The experimental setup was conducted in two dimensions. Initially, datasets were directly input into classifiers prior to the feature selection phase. In the second dimension, datasets underwent feature selection before being processed. The authors employed the WEKA platform and a 10-fold cross-validation method to assess data and to compare outcomes. The performance of the developed model was evaluated based on sensitivity, accuracy, precision, and F-measures. The results highlighted the efficacy of the Farthest First clustering algorithm in software fault prediction, while Bat and Cuckoo algorithms demonstrated superiority compared to other metaheuristic approaches. In software defect datasets, the presence of redundant and irrelevant features can significantly impede the effectiveness of defect prediction models and increase computational complexity. To address this issue, (Alsawalqah, H, et al., 2020) introduced the ReliefF-based clustering [RFC] algorithm, a feature selection (FS) method that identifies and eliminates redundant and irrelevant features. The RFC commences by employing symmetric uncertainty to compute feature correlations and subsequently leverages this data to group features into clusters, employing the k-medoids algorithm. Following this step, it proceeds to choose representative features from each cluster, culminating in the creation of the ultimate feature subset. RFC takes into account both feature correlations and relevance to the targeted class, effectively mitigating that issue of dimensionality and enhancing the performance of SDP. In their experiments, the authors compared the proposed RFC with traditional FS algorithms using nine datasets from the NASA for SDP. Evaluation metrics included the (AUC) area under the curve and F-value. The experiments clearly demonstrated the capability of RFC to significantly enhance SDP performance.

In this research paper (Balogun, A., et al.,2019), they introduced a novel hybrid heterogeneous ensemble method designed for SDP. Heterogeneous ensembles are comprised of classifiers based on various techniques, each with its distinct pros and cons. The primary objective of this researched approach is to construct resilient

and expert heterogeneous machine learning classification models. This approach unfolds in three principal phases: Clustering and Data Segmentation: In the initial phase, they employed a clustering on the training dataset to partition into groups of instances that show some similarity. The determination of the optimal number of clusters in the final model hinges on the G-mean outcomes obtained during the training phase. To mitigate overfitting, they conducted the training using a two-fold cross-validation approach with four settings: 9, 7, 5 or 3 clusters. The optimal number of clusters that yields the highest G-mean value was selected for application in the final testing phase. Classifier Training: The second phase involved training various classifiers based on the groups generated in the 1st phase. Evaluation and Prediction of developed Model: Finally, in the 3rd phase, they assessed the performance of the developed models and employ them to predict instances not represented in the training data. The proposed approach comprises two versions: the first utilizes simple classifiers (k-Nearest Neighbour, Naïve Bayes, and Decision Tree), while the second employs ensemble classifiers (Adaptive Boosting, Bagging, Random Forest and XGBoost). To evaluate the effectiveness of approach, they conducted experiments on 21 publicly available benchmark datasets. Precision, recall, and G-mean served as the evaluation metrics. The outcome of their assessment illustrated the dominance of the ensemble variant when compared to other well-established basic and ensemble classification methods. K-Means is a common and effective clustering technique. Since it can be confused by randomly chosen centroid positions, this method cannot produce optimum results. The central point algorithm was presented in (Annisa, R., Rosiyadi, D., & Riana, D, 2020) as a method for determining the K-Means algorithm's initial centroid value. The point center algorithm is able to count the number of clusters in addition to computing the initial centroid value. This approach pitted ten datasets against each other and relied on choosing variables X and Y to identify the cluster members. Software defect estimation datasets make up 9 of the certified datasets. The results of the suggested approach in the datasets Iris, PC2, PC4, MW1, and KC3 have decreased error results. Some datasets, like PC2, showed the greater Rand Index value. Another experimental finding indicated that, when compared to the straightforward K-Means approach, the suggested algorithm may reduce cluster errors in software defect modules by 12.82%. Since our proposed method outperforms the straightforward K-Means method in terms of accuracy, it may be beneficial for clustering various types of data. The study (Ayon, S. I, 2019) experienced the effectiveness of clustering in the context of SDP. The researchers employed seven distinct clustering techniques: Farthest First, X-Means, K-Means, Sequential Information Bottleneck (SIB), Density-based Clustering (DBC), Hierarchical Clustering (HC) and Expectation Maximization (EM). These techniques were applied to classify eight software datasets sourced from the NASA repository. The experimental results emphasized the well-established utility of clustering techniques as a classification approach, leading to robust predictive performance. By considering average accuracy Farthest First showed the highest performance at 86.16%, closely followed by HC at 85.50%. K-Means displayed a respectable accuracy of 72.33%. Conversely, EM achieved 33.52% accuracy, and X-Means achieved 48.84%, indicating relatively weaker results. SIB and DBC techniques achieved moderate performance at 63% and 71.08%, respectively. Furthermore, a comparison with traditional classification methods such as NB, k-Nearest Neighbor and DT revealed that certain clustering-based classification techniques, namely Farthest First and Hierarchical Clustering Techniques, outperformed some standard classification algorithms. Consequently, clustering-based classification emerges as a viable alternative to traditional methods in SDP, providing robust predictive capabilities while eliminating the need for training predictive models and utilizing annotated datasets during model development. As a result, SDP models developed using clustering-based classification techniques can be easily transferred across projects, as they do not require model training. This approach streamlines resource allocation during the software development process, enhancing efficiency and resource management.

The research (Tang, S., Huang 2022) delved into the influence of feature selection approaches on software defect prediction through clustering techniques. They examined three distinct clustering techniques: Farthest First, K-Means, and

Density Based Clustering, alongside three techniques i.e. Chi-Square, Information Gain and Clustering Variation. These were applied to software defected datasets taken from the NASA repository. The Farthest First classification via clustering algorithm emerged as the top performer, not only achieving the highest average accuracy of 78.57% but also attaining the highest precision of 0.792 and recall of 0.786% values. In the realm of software defect prediction via clustering methods, the Farthest First model led with an accuracy of 78.57%, followed by K-Means at 75.71%, and Density based clustering at 73.77%, respectively. Additionally, their findings indicated that the application of feature selection methods enhanced the average accuracy of both K-Means and Farthest First clustering techniques. The most effective SDP model was the Farthest-First model, incorporating the Information Gain method, achieving an accuracy of 78.69%, a recall of .788 and a precision of .804. This highlights that classification via clustering methods can yield competitive outcomes when compared to conventional classification approaches, with the added advantage of not necessitating training on labeled datasets, making them suitable for application on unlabeled datasets. In practical scenarios, SDP models often face the challenge of handling highly imbalanced datasets. This imbalance creates difficulties for classifiers in accurately identifying defective instances. One common strategy to tackle this issue involves using oversampling techniques, which aim to balance the distribution of faulty and non-faulty dataset instances by generating new instances classified as defective. However, traditional oversampling methods frequently generate synthetic instances lacking diversity and introducing unnecessary noise into the dataset.In response to this concern, a novel approach was introduced by researchers in a study referred to as (Tang, S., Huang, et al., 2022). Their proposed method, known as KMFOS(Cluster-based Over-sampling with noise filtering), was specifically designed to address the imblalance datasets challenge. Initially, the KMFOS approach group the faulty instances in k clusters. It then creates new defective data records through interpolation between the instances belonging to every pair of clusters. This ensures that the newly generated defective instances are distributed diversely across the defective dataset. To further improve the quality of oversampled data, the researchers extended the cluster-based oversampling process by incorporating the CLNI (Closest List Noise Identification) technique to identify and filter out noisy dataset instances.To assess the effective performance of KMFOS, the researchers conducted a comprehensive series of experiments across 24 projects. They compared KMFOS with various oversampling techniques, including SMOTE, BorderlineSMOTE, ADASYN, ROS (random oversampling), SMOTE + IPF, K-means SMOTE, SMOTE + ENN, and SMOTE + Tomek Links. Additionally, they evaluated KMFOS against other standard methods for addressing class imbalance, such as balance bagging classifier, Instance Hardness Threshold, RUS boost classifier and cost-sensitive approaches. The results unequivocally demonstrated that KMFOS outperforms other oversampling methods and class-imbalance mitigation techniques in terms of achieving higher Recall and balance values. In the study presented in (Ayon, S. I, 2019), the authors introduced a multi-step approach for feature selection and classification. Initially, they employed Genetic Algorithm (GA) to select relevant features from the dataset. Subsequently, they formed feature clusters using PSO. Following this feature preprocessing stage, the researchers trained their model using various Neural Network methods, including Recurrent Neural Network (RNN), Feedforward Neural Network (FNN), Deep Neural Network (DNN) and Artificial Neural Network (ANN). To evaluate the model's performance, they computed several evaluation metrics, including sensitivity, accuracy, specificity, negative prediction value, precision, F1 score, and Matthews correlation coefficient. To assess the proposed approach's effectiveness, the authors conducted experiments on five distinct datasets, namely: PC1, KC1, CM1, JM1, and KC2. To ensure robust evaluation, they implemented 10-fold cross-validation . Remarkably, results revealed that the DNN consistently achieved accuracy levels exceeding 90% across all datasets. Additionally, the other three methods—FNN, RNN, and ANN—also demonstrated strong performance, yielding commendable accuracy results.

An approach for software defect prediction based on clustering-based undersampling and ANN is presented in research (Zhang, S., Jiang, S., & Yan, Y. 2023). First, a subset of the majority samples is

chosen using clustering-based under-sampling, and this subset is joined with the minority samples to create a balanced data set. Second, using the balanced data set produced, an ANN model is created and trained. Ten nodes in a single hidden layer made up the created ANN model. Additionally, softmax function is used which is the output layer's transfer function, while log-sigmoidal was the transfer function of hidden layer. Adam is employed to instruct the ANN. Adam is a stochastic objective function optimization that uses gradients. To determine the number of majority samples that produces the best performance metrics, a sensitivity analysis is also done. Accuracy, PF (Probability of False Alarm or FPR), PD (Probability of Detection or T PR), F1-measure, ROC, and Balance are some of the performance metrics employed. The results demonstrate a high degree of prediction accuracy for the detection of defective modules while preserving the capability of defect-free module detection. Most defect prediction methodologies require a substantial amount of labeled data for effective training, typically on modules within the same project. Such approaches are commonly referred to as "within project defect prediction" (WPDP). However, WPDP models may encounter challenges when confronted with limited training data. To address this issue, researchers have introduced cross project defect prediction and unsupervised machine learning techniques. CPDP models, as opposed to WPDP, utilize training data sourced from different projects for the purpose of forecasting defect proneness in modules within a particular project. Nevertheless, CPDP's performance is notably sensitive to variations in data distribution across different projects. In (Xiaolong, X. U, et al., 2021) they proposed a novel approach known as Cluster Ensembles and Labeling (CEL) for unlabeled datasets defect prediction. Instead of performing clustering on a single dataset, CEL employs a methodology involving the generation of multiple data partitions. Cluster algorithms are subsequently applied to these partitions, and the results from these various clusters are then combined. To assess the effectiveness of CEL, a series of experiments were conducted using fifteen(15) projects obtained from three(3) distinct data repositories, specifically , AEEEM, Relink, and PROMISE. The experimental outcomes reveal that CEL exhibits superior prediction performance compared to the exceptional unsupervised learning model CLA. Additionally, a comparative analysis was performed between CEL and several supervised learning models, including Decision Tree , Logistic Regression and Random Forest employing evaluation metrics like precision, recall, and F-measure. In conclusion, the research findings consistently demonstrate that CEL outperforms the CLA model across a range of experimental datasets. The researchers in (Almayyan, W, 2021). introduced an innovative approach called Feature Selection using Clusters of Hybrid-data. Their primary objective was the minimization of the feature distributions dissimilarity in the context of CPDP (Cross Project Defect Prediction) between the source and target project datasets. The proposed method consists of two stages. Initially, it utilizes the DPC (Density-Based Clustering) to partition the real dataset into number of clusters. Subsequently, three distinct ranking strategies, namely Similarity of Feature Distributions (SFD), Local Density of Features (LDF), and Feature-Class Relevance (FCR), are proposed to select features from each cluster. These approaches do not aim to map features but rather offer guidance in selecting relevant software metrics within the context of CPDP.To evaluate the effectiveness of FeSCH, a series of experiments were conducted using the AEEEM dataset. These experiments included a comparative analysis with alternative methods and a thorough examination of its design choices. To assess FeSCH's performance across different classifiers, a comparison was carried out with three widely used classifiers: Random Forest, Logistic Regression and Naive Bayes. The conclusive findings demonstrate that FeSCH consistently outperforms three methods (ALL,WPDP and TCA+) in various scenarios, maintaining its superiority irrespective of the chosen classifiers.

REFERENCES

Thirumoorthy, K., & Britto, J. J. J. (2022). A clustering approach for software defect prediction using hybrid social mimic optimization algorithm. Computing, 104(12), 2605-2633.

Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2023). Software Defect Prediction Analysis Using Machine Learning Techniques. Sustainability, 15(6), 5517.

Zhang, S., Jiang, S., & Yan, Y. (2023). A Software Defect Prediction Approach Based on Hybrid Feature

Dimensionality Reduction. Scientific Programming, 2023.

Tang, S., Huang, S., Liu, E., Yao, Y., Wu, K., & Ji, H. (2022). Tsbagging: A Novel Cross-Project Software Defect Prediction Algorithm Based on Semisupervised Clustering. Scientific Programming, 2022.

Almayyan, W. (2021). Towards predicting software defects with clustering techniques. International Journal of Artificial Intelligence and Applications (IJAIA), 12(1).

Xiaolong, X. U., Wen, C. H. E. N., & Xinheng, W. A. N. G. (2021). RFC: a feature selection algorithm for software defect prediction. Journal of Systems Engineering and Electronics, 32(2), 389-398.

Alsawalqah, H., Hijazi, N., Eshtay, M., Faris, H., Radaideh, A. A., Aljarah, I., & Alshamaileh, Y. (2020). Software defect prediction using heterogeneous ensemble classification based on segmented patterns. Applied Sciences, 10(5), 1745.

Annisa, R., Rosiyadi, D., & Riana, D. (2020). Improved point center algorithm for k-means clustering to increase software defect prediction. Int. J. Adv. Intell. Informatics, 6(3), 328-339.

Balogun, A., Oladele, R., Mojeed, H., Amin-Balogun, B., Adeyemo, V. E., & Aro, T. O. (2019). Performance analysis of selected clustering techniques for software defects prediction.

Usman-Hamza, F. E., Atte, A. F., Balogun, A. O., Mojeed, H. A., Bajeh, A. O., & Adeyemo, V. E. (2019). Impact of feature selection on classification via clustering techniques in software defect prediction. Journal of Computer Science and Its Application, 26(1).

Gong, L., Jiang, S., & Jiang, L. (2019). Tackling class imbalance problem in software defect prediction through cluster-based over-sampling with filtering. IEEE Access, 7, 145725-145737.

Ayon, S. I. (2019, May). Neural network based software defect prediction using genetic algorithm and particle swarm optimization. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (pp. 1-4). IEEE.

Yang, Y., Yang, J., & Qian, H. (2018, March). Defect prediction by using cluster ensembles. In 2018 tenth international conference on advanced computational intelligence (ICACI) (pp. 631-636). IEEE.

Ni, C., Liu, W., Gu, Q., Chen, X., & Chen, D. (2017, July). FeSCH: a feature selection method using clusters of hybrid-data for cross-project defect prediction. In 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 51-56). IEEE.

Sharma, T., Jatain, A., Bhaskar, S., & Pabreja, K. (2023). Ensemble Machine Learning Paradigms in Software Defect Prediction. Procedia Computer Science, 218, 199-209.

Mafarja, M., Thaher, T., Al-Betar, M. A., Too, J., Awadallah, M. A., Abu Doush, I., & Turabieh, H. (2023). Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning. Applied Intelligence, 1-43.

Bowes, D., Hall, T., & Petrić, J. (2018). Software defect prediction: do different classifiers find the same defects?. Software Quality Journal, 26, 525-552.

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.

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.

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).