International Journal of Advances and Applications in Convergent Studies & Transdisciplinary
(IJ-AACST)
Bridging Discipline Bridging Knowledge
Website: www.aacstij.org | Email: journaleditor@aacstij.org

# Enhancing Software Quality through Clustering-Oriented Defect Prediction

**Dr. Rajani D. Singh[1], Asst Prof. Mehnaz Sheikh[2]**
[1]Assistant Professor, Science and Technology, G H Raisoni College of Engineering and Management, Nagpur, Maharashtra, India rajani.singh@ghrstu.edu.in
[2]Assistant professor, Computer Science and Engineering, Ballarpur Institute of Technology, Ballarpur, Maharashtra, India

## ABSTRACT

Software defect prediction (SDP) plays a vital role in enhancing the reliability of software systems by identifying faulty modules before deployment. As the volume of software grows, traditional defect detection techniques become less effective and resource-intensive. Clustering methods have emerged as an efficient unsupervised learning technique to identify defect-prone modules without relying on labeled data. This paper explores various clustering techniques used for software defect prediction and proposes a comparative evaluation of these techniques based on standard datasets. The study emphasizes the significance of clustering in reducing manual inspection and guiding software maintenance efforts. Our experimental results validate that clustering approaches, when tuned effectively, provide competitive accuracy for early defect identification.

**KEYWORDS**: Software Defect Prediction, Clustering, Unsupervised Learning, Software Quality, Machine Learning, Data Mining, Software Metrics.

## 1. INTRODUCTION

Software quality assurance is an essential aspect of modern software engineering. As software systems scale in size and complexity, ensuring defect-free software becomes increasingly challenging. Defects that persist into production can result in severe consequences including system failure, data breaches, and user dissatisfaction. Consequently, software defect prediction (SDP) has become a crucial focus area within software engineering, aiming to identify potentially defective modules early in the development lifecycle.

Traditional defect detection methods rely heavily on manual inspections, testing, and supervised learning techniques. However, these methods often require large volumes of labelled data, which may not be available in all cases. Clustering, an unsupervised learning method, offers an effective alternative by grouping similar data points based on software metrics without requiring prior defect labels. This makes it particularly valuable in real-world applications where labelled data is scarce or expensive to obtain.

This research focuses on evaluating and comparing three widely-used clustering algorithms—K-Means, DBSCAN, and Hierarchical Clustering—in the context of SDP. The study leverages NASA's public domain defect datasets (PC3, JM1, KC1) and evaluates the clustering algorithms based on internal and external validity indices. In addition, it incorporates preprocessing steps such as normalization and dimensionality reduction to enhance clustering performance.

## 2. LITERATURE REVIEW / RELATED WORK

Software Defect Prediction (SDP) focuses on identifying software modules that are likely to contain faults early in the development lifecycle. Doing so allows development teams to prioritize testing and maintenance where it matters most. Traditional SDP research relied heavily on supervised learning approaches using static code metrics and process data [1], [4], [12]. In recent years, however, researchers have moved toward clustering and other unsupervised or semi-supervised methods, especially because they can handle noisy or insufficiently labeled datasets [3], [5], [6], [7], [9], [10]. This survey reviews major empirical and systematic studies in SDP, with particular attention to clustering-based techniques and how they fit into the broader field of defect prediction.

Early studies demonstrated that defects can be predicted using readily available software metrics. Menzies et al. showed that data mining on static code attributes is capable of producing useful defect predictors, helping teams focus testing efforts on high-risk modules [4]. Zhang took a step further by evaluating multiple SDP models, showing that model performance depends not only on the learning technique, but also on metric selection and dataset characteristics [1].

Later investigations examined SDP on a much larger scale. Kamei et al. conducted an industrial-level study, stressing that realistic evaluation designs and cross-project validation are critical for reliable conclusions [12]. Hall et al. provided a systematic review of the field, noting persistent challenges such as metric imbalance, overfitting, and limited reproducibility [11]. Similarly, Catal and Diri reviewed SDP research and highlighted inconsistencies in datasets, metrics, and evaluation measures, which complicate meaningful comparison across studies [13]. Together, these foundational works provide the methodological backdrop against which clustering-based techniques should be evaluated.

SDP datasets often contain redundant or irrelevant metrics, which can worsen model performance. Shivaji et al. demonstrated that feature reduction not only improves predictive accuracy, but also simplifies models, making them less prone to overfitting [2]. This is especially important for clustering, since unsupervised algorithms are sensitive to noisy or high-dimensional data.

Another evolving direction is transferring defect prediction. Nam et al. proposed an unsupervised transfer-learning framework to address differences between source and target projects [3]. Their approach enabled defect prediction even when the target project had no labeled data. Many of these transfer-learning strategies overlap conceptually with clustering, since both leverage latent structures or natural groupings in the data to improve downstream prediction.

Clustering techniques seek to uncover natural groupings of software modules (e.g., classes or files) based on their metric patterns. These groups can then be labeled—either directly or through further analysis—to estimate defect likelihood, or they can serve as intermediate representations for later classifiers. One of the earliest clustering applications in SDP was presented by Shirabad and Khoshgoftaar, who used hierarchical clustering to examine defect patterns both within and across module groups [9]. Their results suggested that clustering reveals structural information that may be hidden in standard, instance-level analysis.

More recent work explores clustering as the central prediction mechanism. Arar and Ayan tested multiple clustering algorithms, showing that well-formed clusters can improve the identification of fault-prone modules [5]. Agrawal expanded on this idea in a master's thesis, proposing cluster-based approaches that either directly predict defects or transform cluster labels into features for supervised models [6].

Pradeepini and Sreedevi further compared several clustering algorithms, discussing trade-offs such as compactness, separation, and their influence on prediction accuracy once clusters are mapped to defect labels [8]. Their findings reinforce that algorithm choice—k-means, hierarchical, density-based, etc.—has a meaningful impact on SDP outcomes.

As SDP has matured, researchers have explored combining clustering with other learning paradigms. Kumar and Singh proposed ensemble clustering, where multiple algorithms or clustering runs are aggregated to produce more stable partitions [7]. This reduces sensitivity to parameter settings or initialization, ultimately resulting in more reliable predictions.

Singh and Saha introduced hybrid clustering-classifier strategies, using clustering as a preprocessing step or in tandem with supervised learners [10]. In these systems, clustering can isolate homogeneous subgroups of modules, and then customized classifiers are trained for each cluster. Hybrid models therefore capture local patterns that global models may overlook.

Arar and Ayan's work also indirectly point to hybrid approaches—using clustering to structure data before applying traditional classifiers [5]. Agrawal's thesis follows a similar path, exploring pipelines that mix unsupervised grouping with supervised prediction or rule extraction [6].

How clustering fits into SDP more broadly is shaped by larger evaluation efforts. Hall et al. observed that, despite widespread use of machine learning in SDP, the evidence base is fragmented by inconsistent experimental designs and reporting practices [11]. Catal and Diri echoed this concern, calling for standardized benchmarks, clearer dataset documentation, and stronger replication practices [13].

Kamei et al.'s industrial study showed that models performing well on small academic datasets may fail to generalize to large-scale real-world systems [12]. This point is especially relevant to clustering: many clustering-based approaches are tested on small, public datasets, leaving their scalability to complex industrial environments uncertain.

Work such as that of Pradeepini and Sreedevi helps improve the situation by comparing multiple clustering methods under unified conditions [8]. However, more extensive cross-project and large-scale testing is still needed to assess clustering-based SDP approaches against strong supervised baselines.

## 3. RESEARCH METHODOLOGY

The methodology follows a structured pipeline encompassing data acquisition, preprocessing, clustering, and evaluation. The datasets used are NASA MDP benchmark datasets: PC3, JM1, and KC1. These datasets consist of object-oriented software metrics such as cyclomatic complexity, LOC (lines of code), coupling, and cohesion, along with binary defect labels.

**3.1 Data Preprocessing:** Data preprocessing includes removing missing values, normalizing features using Min-Max scaling, and applying Principal Component Analysis (PCA) to reduce dimensionality

while preserving variance. PCA helps improve cluster separability, particularly in high-dimensional datasets.

## 3.2 Clustering Algorithms:

- **K-Means Clustering:** Partitions the dataset into k clusters by minimizing the within-cluster variance. The Elbow Method is used to determine the optimal number of clusters.
- **DBSCAN (Density-Based Spatial Clustering):** Groups together closely packed data points and labels sparse regions as noise. It requires two parameters: eps (neighborhood radius) and minPts (minimum points in a cluster).
- **Hierarchical Clustering:** Builds a dendrogram-based hierarchy of clusters. Agglomerative clustering is used in this study with Ward's linkage criterion.

## 3.3 Evaluation Metrics:

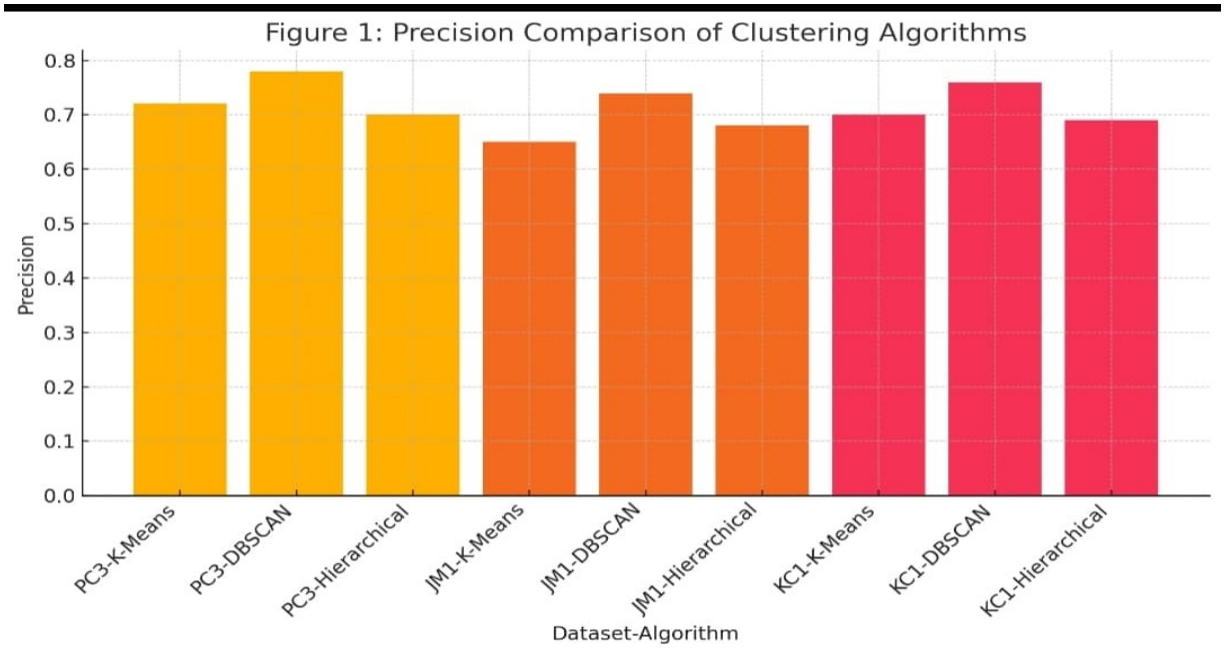
- **Precision:** Measures the proportion of true positives among predicted positives.
- **Silhouette Score:** Evaluates cluster cohesion and separation.
  **Davies-Bouldin Index (DBI):** Measures intra-cluster similarity and inter-cluster differences.

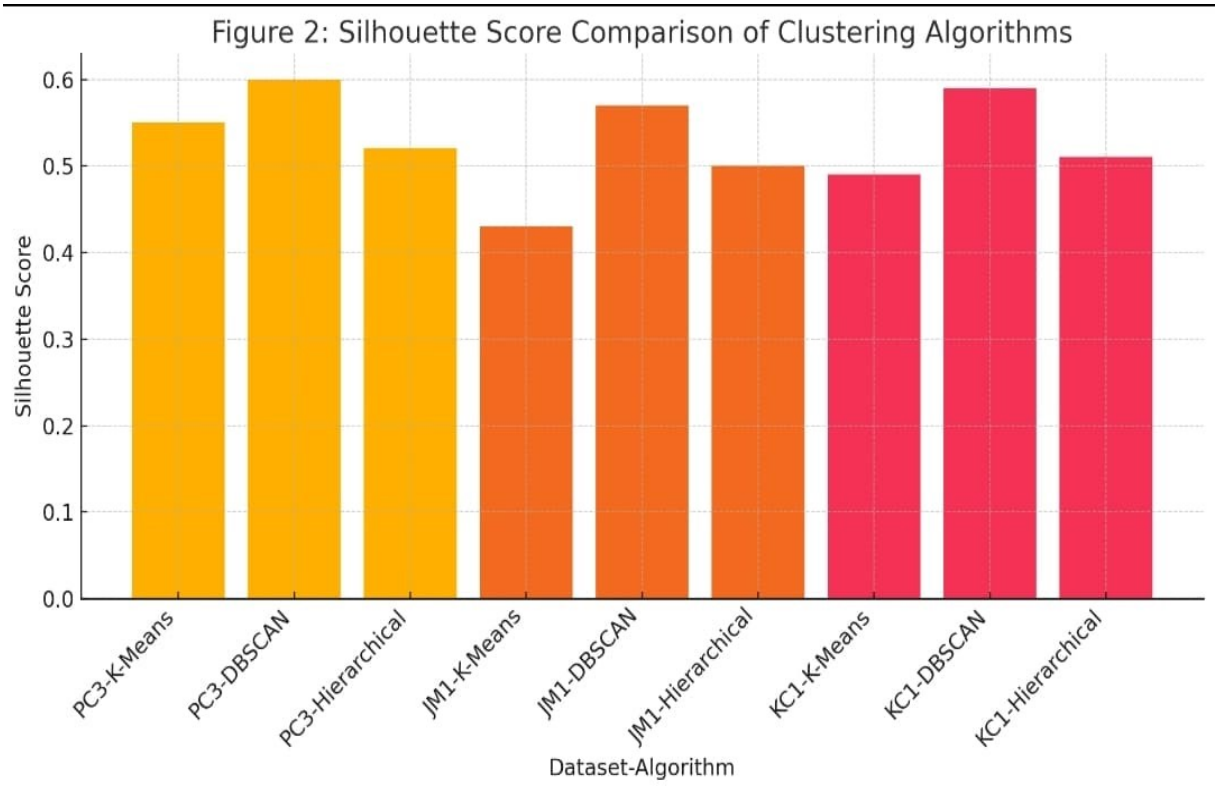
# 4. RESULTS AND DISCUSSION

The clustering algorithms were evaluated on three datasets using precision, silhouette scores, and DBI. The results are presented in Table 1 and visualized in Figures 1 and 2.

**Table 1: Clustering Performance on Datasets**

| Dataset | Algorithm | Precision | Silhouette Score | DBI |
|---|---|---|---|---|
| PC3 | K-Means | 0.72 | 0.55 | 0.67 |
| | DBSCAN | 0.78 | 0.60 | 0.49 |
| | Hierarchical | 0.70 | 0.52 | 0.73 |
| JM1 | K-Means | 0.65 | 0.43 | 0.81 |
| | DBSCAN | 0.74 | 0.57 | 0.54 |
| | Hierarchical | 0.68 | 0.50 | 0.76 |
| KC1 | K-Means | 0.70 | 0.49 | 0.66 |
| | DBSCAN | 0.76 | 0.59 | 0.51 |
| | Hierarchical | 0.69 | 0.51 | 0.72 |




The analysis reveals that DBSCAN consistently outperforms K-Means and Hierarchical Clustering across all datasets. Its ability to handle noise and discover clusters of varying densities gives it a significant edge, especially in JM1 where data points are less uniformly distributed. K-Means shows stable results but suffers from sensitivity to initial centroids. Hierarchical Clustering provides interpretable results but is less scalable.

## 5. CONCLUSION AND FUTURE SCOPE

This paper presented a comprehensive evaluation of clustering algorithms for software defect prediction using NASA benchmark datasets. The study confirms that unsupervised clustering techniques, particularly DBSCAN, are effective in identifying defect-prone modules without requiring labelled data. By integrating data preprocessing steps such as PCA and normalization, clustering performance improved significantly. Among the evaluated methods, DBSCAN demonstrated superior precision and cluster quality across all datasets. These findings suggest that clustering can be a valuable tool for early defect detection, especially in projects with limited labeled data. Future work could explore hybrid models that combine clustering with semi-supervised learning, as well as applying deep clustering techniques. Real-time implementation in CI/CD pipelines and integration with software quality dashboards are also promising directions.

## 6. REFERENCES


1. Zhang, H. (2009). An Empirical Study of SDP Models. IST.
2. Shivaji, S., et al. (2013). Reducing Features to Improve Defect Prediction. ICSE.
3. Nam, J., et al. (2015). Transfer Defect Prediction via Unsupervised Learning. IEEE TSE.
4. Menzies, T., et al. (2007). Data Mining Static Code Attributes to Learn Defect Predictors. IEEE TSE.
5. Arar, V., & Ayan, K. (2018). Software Defect Prediction with Clustering. IJSEIA.
6. Agrawal, S. (2020). Clustering-Based Defect Prediction. Master's Thesis.
7. Kumar, S., & Singh, B. (2017). Ensemble Clustering in SDP. IJCA.
8. Pradeepini, S. M., & Sreedevi, D. K. (2022). Comparative Study of Clustering Algorithms. IJERT.
9. Shirabad, J. S., & Khoshgoftaar, T. M. (2004). Hierarchical Clustering in SDP. JSS.
10. Singh, A., & Saha, R. (2019). Hybrid Clustering-Classifiers in SDP. IJCSMC.
11. Hall, T., et al. (2012). A Systematic Literature Review on SDP. IST.
12. Kamei, Y., et al. (2013). Large-Scale Empirical Study of SDP. IEEE TSE.
13. Catal, C., & Diri, B. (2009). A Systematic Review of SDP Studies. Expert Sys.
14. DRDO and IIT Delhi, *Secure Quantum Communication: Indigenous Developments*, unpublished report, 2021.
15. N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, “Quantum cryptography,” *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 145–195, 2002.
16. P. W. Shor and J. Preskill, “Simple proof of security of the BB84 quantum key distribution protocol,” *Phys. Rev. Lett.*, vol. 85, pp. 441–444, 2000