Journal of Information Systems Engineering and Management
2025, 10(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Reducing Redundancy in Software Testing: A K-Means Clustering Approach to Test Case Minimization

Sanjay Sharma[1], Jitendra Choudhary[2]

[1]*Schoool of Computers and Electronics, IPS Academy, Indore,*

[1,2]*Department of Computer Science, Medicaps University, Indore*

[1]*sanjaysharmaips61@gmail.com*, [2]*jitendra.choudhary@medicaps.ac.in*

ARTICLE INFO

Received: 07 Oct 2024

Revised: 05 Dec 2024

Accepted: 19 Dec 2024

ABSTRACT

This study explores a clustering-based approach using k-means clustering to optimize test case minimization, a crucial aspect of enhancing software testing efficiency. Traditional methods often fail to cope with the complexity and scale of modern systems, leading to excessive redundancy and resource consumption. By employing k-means clustering, we effectively group similar test cases into clusters, enabling a reduction in the number of test cases while maintaining adequate test coverage. This approach offers several benefits, including reduced testing time, improved resource utilization, and enhanced fault detection capabilities. Our method ensures that critical areas of software functionality are thoroughly tested, even with fewer test cases, leading to cost savings and efficient resource allocation. The results demonstrate that this clustering-based test case minimization technique is both scalable and practical, making it suitable for large-scale software testing environments

Keywords: Software testing, optimization, clustering, resource utilization, algorithms, test cases

INTRODUCTION

Software testing is one of the critical phases of the development lifecycle, validating the reliability and functionality of systems. However, the whole process is often challenged by large, complex datasets with resource constraints. Conventional test case minimization techniques face such difficulties more often, resulting in inefficiencies like redundant test cases, increased testing times, and greater resource consumption. As a response, this paper presents a clustering-based approach using the k-means clustering technique toward the minimization of test cases. Grouping similar test cases under one set of clusters has in fact reduced redundancy while ensuring adequate coverage over the critical functions of the software. This alone saves testing time and improved resource utilization, coupled with the enhancement of the fault-detection capability. The k-means clustering method addresses the scale and complexity of the modern software systems, making it an effective and practical solution for the large projects requiring effective and efficient testing strategies.

The rest of the paper is organized as follows: Section II describes the methodology adopted in this study, detailing the clustering-based approach for test case minimization, the experimental setup, and the technical implementation. Section III focuses on the results and discussion, analyzing the performance of the proposed approach using key metrics such as accuracy, precision, recall, and fault detection efficiency. Finally, Section IV concludes the paper by summarizing the key findings, highlighting the contributions of the study, and providing directions for future research.

K-means is one of the most used clustering algorithms in practice; however, it is often suffering from the a priori choice of the number of clusters and initializations. Thus, one can say that its unsupervised nature is somewhat limited. In [1], an unsupervised k-means (U-k-means) clustering algorithm was proposed that automatically determines the optimal number of clusters without any initialization or parameter selection, resulting in big improvements in clustering performance. A second contribution was the K-means Optimizer (KO), developed using

the k-means algorithm to generate the centroid vector and combined movement heuristics to balance exploration and exploitation in optimization problems. The new approach performed very well in benchmark and structural damage identification problems, outperforming other optimization algorithms [2]. Another study applied k-means and k-medoids clustering in order to improve test case prioritization in regression testing. This improved the efficiency of fault detection compared to random prioritization methods [3]. Another modification of noise-based k-means clustering algorithm has been done towards establishing solutions for urban congestion through the automatic choice of number of clusters and initialization of cluster centers, which performed better in clustering for multiple urban datasets [4].K-means clustering algorithm is one of the popular algorithms in most clustering tasks that are often used because of their simplicity and efficiency, but it is limited. For example, it is sensitive to the centers' initialization and finding the optimal number of clusters. A suggested algorithm was a kind of iterative improvement of k-means by eliminating and splitting clusters, which led to significantly improved and even better accuracy than k-means++ [5]. Another study applied a hybrid of k-means algorithm with the elbow method to select the number of clusters. It efficiently improved the performance of k-means in SME customer mapping by choosing the most appropriate number of clusters corresponding to the lowest values of SSE [6]. Another research attempt has been made into a data-driven approach for automatic finding of optimal numbers of clusters for k-means, based on cluster symmetry analysis, which was tested over real-world datasets such as satellite images and indicated the achievement of improved accuracy and lesser values of root mean square error in comparison with the traditional methods [7]. Such progress in k-means clustering testifies to the continuous attempt to overcome its inherent lack and thus apply it to each possible domain: from ordinary customer profiling to the analysis of data from satellites.The multiple explorations of k-means clustering and its variants across different domains highlight strength in versatility as well as disadvantages, including sensitivity to initialization and the determination of the optimal number of clusters. These findings are somehow combined by comparing with a Table 1 below summarizing the key studies, their methodologies, datasets, evaluation metrics, and results, offering a broader view of the advancements in k-means clustering and applications.

Table 1: Comparative Analysis of K-Means Clustering Variants and Applications across Different Domains

Study	Method/Algorithm	Focus Area/Problem Addressed	Dataset Used	Evaluation Metrics	Key Techniques/Approach	Key Findings/Results
[8]	K-Means with SVM	Data stream mining, concept drift	Testbed, NSL-KDD, CIDDS-2017	Accuracy, Precision, Recall, F1 Score, Kappa Statistics	Sliding window, drift detection, SVM retraining	Achieved improved accuracy (93.52%, 99.80%, 91.33%) and precision with SVM using drift detection techniques.
[9]	K-Means with Elbow and Silhouette	Cluster determination, distance metrics	Synthetic dataset	SSE, Elbow method, Silhouette score	Comparison of Manhattan, Euclidean, and Minkowski distances	Manhattan distance showed the most variance; K determination remains challenging.
[10]	Maxmin and K-Means++	Improving K-Means initialization	Clustering benchmark	Accuracy, Cluster Quality (Error Rates)	Furthest point heuristic (Maxmin), Repeated initialization	Maxmin reduces erroneous clusters from 15% to 6%; repeated K-Means reduces errors to 1%.
[11]	K-Means in Education Data	Student performance analysis	Educational datasets	Cluster Performance (Prediction Accuracy)	K-Means, Decision Trees	K-Means effectively groups students based on performance, improving educational outcomes.
[12]	K-Means and K-Medoids with Transformation	Medical data clustering	Local medical data	Accuracy, Cluster Robustness	Yeo-Johnson transformation, SEV distance function	Transformed data significantly improves clustering accuracy; Yeo-Johnson method is best for accuracy.
[13]	KNSMOTE with K-Means	Imbalanced medical data	8 UCI datasets	Sensitivity, Specificity	Synthetic Minority Oversampling Technique (SMOTE) with K-Means	KNSMOTE improves sensitivity (99.84%) and specificity (99.56%) in medical

						datasets.
[14]	Multiple Kernel Clustering (MKC)	Incomplete kernel matrices	13 Benchmark datasets	Clustering Accuracy, Time Complexity	Joint imputation and clustering, kernel subset selection	Superior clustering performance, particularly with high missing data ratios.
[15]	Hierarchical K-Means and Cubist Algorithm (HKM-CA)	Blast-induced Peak Particle Velocity (PPV) prediction	Open-pit mine data	RMSE, R^2, MAE	Hybrid HKM-CA model, Cubist model for prediction	HKM-CA outperforms other models with RMSE of 0.475 and R^2 of 0.995.
[16]	Joint Clustering and Representation Learning	Clustering and representation learning	Various datasets	Clustering Accuracy, Data Representation	Continuous reparametrization of clustering objective	Improved clustering performance by jointly learning data representations.
[17]	K-Means based Co-Clustering (kCC)	Enhancing K-Means clustering	Standard datasets	SSE, Cluster Assignment Accuracy	Co-clustering, neighborhood walk statistics	kCC significantly improves clustering accuracy over baseline K-Means.
[18]	Clustering and Genetic Algorithm	Regression testing time minimization	Software test suites	Execution Time Reduction	Test suite reduction, parallel execution	Reduced regression testing time by 75%, preserving code coverage.
[19]	ATM (AST-based Test Case Minimizer)	Test case minimization using AST similarity	Dataset of 16 Java projects	Fault Detection Rate, Execution Time	AST-based similarity measures, genetic algorithms	Achieved a 0.82 average fault detection rate, outperforming FAST-R (0.61) and random minimization (0.52) with 50% test cases.
[20]	Cluster-Based Adaptive Test Case Prioritization	Real-time adaptive TCP	Java programs (1 open-source, 3 industrial-grade)	APFD Improvement Rate	Pre-prioritization, adaptive adjustment algorithm	Median APFD improvement of 17.08% (step=2), significantly outperforming other TCP techniques.
[21]	Clue (Test Reduction for SPL Systems)	Test reduction for software product lines	Dataset of 6 SPL systems	Fault Detection, Fault Localization	Clustering tests, prioritization by feature interactions	Reduced testing effort by up to 88% while maintaining fault localization performance; detected most bugs using 50% of the original test suites.

Wireless Sensor Networks (WSNs) consist of distributed sensor nodes that sense and disseminate environmental data with problems such as redundant broadcast storms, energy inefficiency, and security vulnerabilities. In this respect, a novel Cluster-based Secured Data dissemination Protocol (CSDP) is proposed based on energy-efficient and secure communication. CSDP includes digital signature authentication, trust-based security, encryption for key management, and an Intelligent Fuzzy-based Unequal Clustering algorithm for effective clustering and intruder detection [22]. This ensures strong security, efficient routing, and minimal energy usage as it uses cryptographic key generation, computation of the trust score, and the prevention of malicious nodes. CSDP was tested using NS2-based simulations for packet delivery, network throughput, and delay. Moreover, optimizations to kernel-based clustering by the selection of an optimal subset of the kernel and redundancy reduction improve the clustering efficiency and performance, which was validated through extensive experiments [23], [24].

METHODOLOGY

The study is about optimization of software testing in order to minimize redundant test cases so that no redundancy occurs in their usage with saving of time as well as cost incurred while never compromising on the effectiveness of fault detection. The paper evaluates traditional optimization approaches, such as genetic algorithms, and the research study introduces K-Means clustering, which might be used for better minimization of test cases. K-Means is the approach proposed for handling large suites more efficiently when scalability and complex features of the test

cases pose problems. Metrics in this regard would be test suite reduction, code coverage, and execution time. Thus, it will give an insight into how the approach can enhance software testing.This section reports the technical approach on effective software testing using a K-Means clustering-based test case minimization algorithm. In contrast, the study focuses on machine learning based source code analysis techniques rather than traditional methods. The setup and execution of the original and minimized test suites are conducted in a test environment followed by comparison using key performance metrics such as test execution time, rate of fault detection, and other factors affecting resource usage. The proposed method systematically manages redundancy reduction, test selection, and performance evaluation to improve software quality while lowering the time and resources for testing.

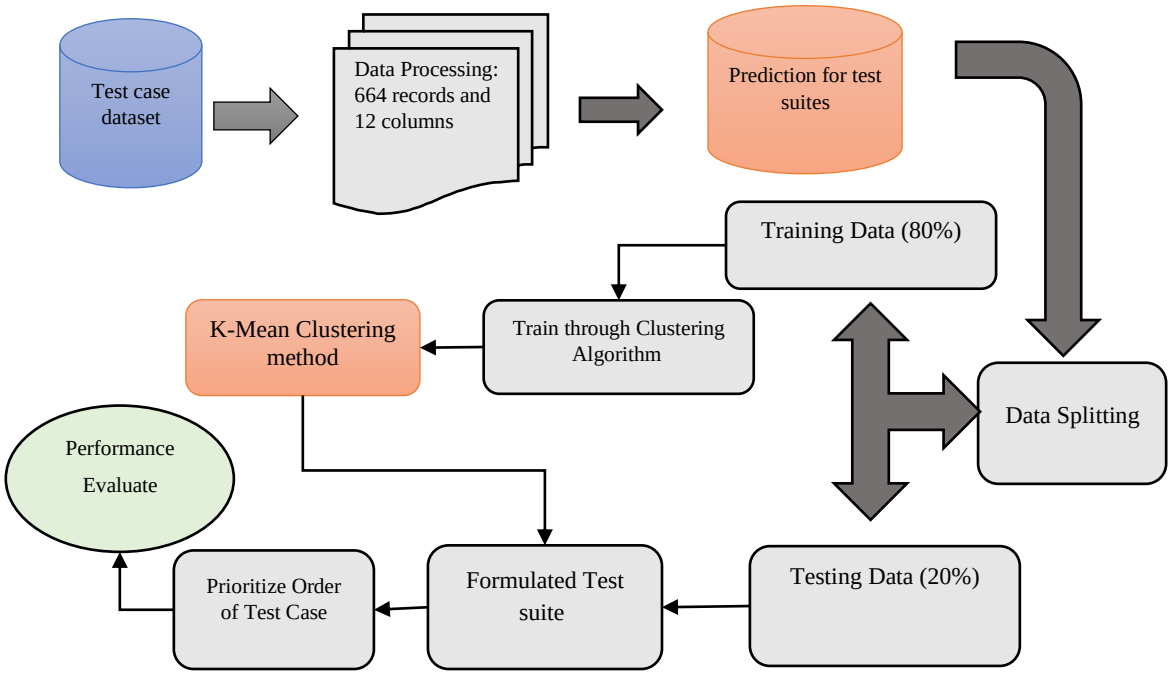

Fig.1Comprehensive flowchart of the proposed model

A set of test cases is designed for the newly developed model in addition to existing general test cases. The goal is to determine the optimal order for applying these test cases to rapidly detect faults through efficient pre-processing techniques. To establish this order, the test case suite is predicted and split into training and testing sets. Clustering is then applied to the training data based on factors inherent in the test suite, prioritizing the test cases according to performance evaluation. The proposed algorithm, begins with data collection, including regression and edge test cases, followed by generating test cases using combinatorial test design as shown in Fig.1. Pre-processing the data involves structuring 664 attributes across 12 columns for clustering. Next, the data is split into 80% for training and 20% for testing, after which the K-Means clustering algorithm groups the test cases based on Clustering Criteria (CC attributes), such as accuracy, precision, recall, and F1-score. These metrics are critical for evaluating clustering effectiveness and ensuring optimized grouping of test cases for enhanced testing efficiency.

The methodology involves the publicly available N_BaIoT dataset, taken from Kaggle to further enhance analysis and then estimate the potential for automating intrusion detection in IoT networks. Four releases of test cases for AuthZ microservices are utilized in this study as test data to validate the proof of concept. AuthZ, an authorization microservice within the HP Cloud Print Platform, is specifically implemented using the OAuth2 standard. While OAuth2 is a general protocol for authorization, AuthZ represents a tailored application of this protocol, designed to manage access control across systems and microservices. Currently, testing of selected tests happens in the HP Cloud Print Platform. This process will be followed by a release manifest, versioned items, configuration settings, and descriptions of issues in each release, using tools such as managing the projects in JIRA, a Service Functionality Mapping File for reporting impacted functionalities, and Test Rail for test management. The SME will refer to the manifest of the release, refer to the reports in JIRA for stories and bugs, and then use the functionality mapping to determine the affected areas. The SME then selects the appropriate test cases from Test Rail by using

the data available and aggregated. The dataset includes columns such as unique identifiers for each record, release IDs, test case types, descriptions, error-prone test cases, automation status, defects, JIRA bug IDs, bug descriptions, GIT commit messages, and binary classifications for selecting test cases.

RESULTS AND DISCUSSION

The machine learning model was trained on Google Colab using TensorFlow and Keras, with 80% of the dataset for training and 20% for testing. It trained for over 100 epochs using the Adam optimizer with a 0.0001 learning rate on a Tesla P100-PCIE GPU. The main contributions of this work include investigating the efficiency of clustering algorithms for test case minimization and demonstrating the effectiveness of K-Mean Clustering in identifying Centroids using the Silhouette Visualizer. It also provides key parameter values for achieving high detection accuracy with K-Means. Additionally, the research advances the implementation of the K-Mean algorithm using Google TensorFlow and Python. The heat-map given in **Fig.2** plots the correlation matrix for a set of variables. In this matrix, each cell is the correlation coefficient between two different variables. These coefficients range from -1 (dark blue) for a strong negative correlation to 1, which represents a strong positive correlation, bright green and yellow. The diagonal is all perfect correlations at 1, as any variable is perfectly correlated with itself. Important correlations comprise a strong negative correlation of Id and ReleaseID (-0.97), a moderate positive correlation between Type of Test Case and Any Defect (0.26) and between Automation Status and Error Prone Test Cases (0.30).

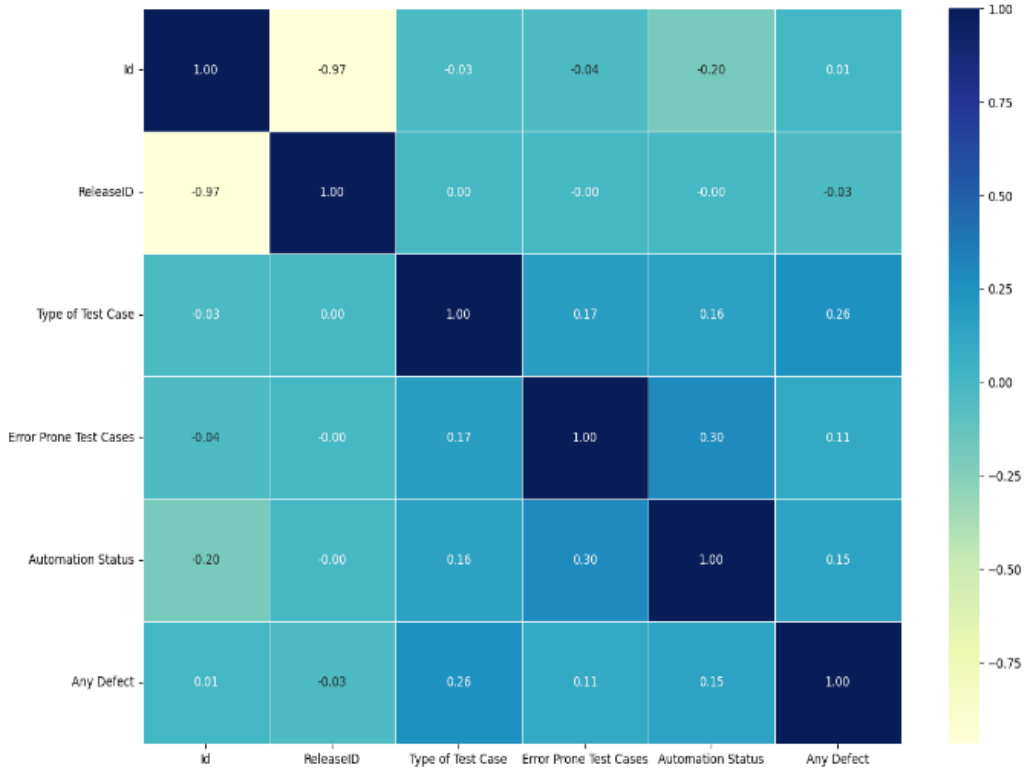

Fig.2: Correlation heat map for evaluating accuracy, sensitivity, and specificity in test case minimization

The correlation heatmap, highlighting relationships that suggest SMEs/Test Engineers may not currently be considering certain features or columns when selecting test cases. However, we strongly recommend that these features be taken into account, as they directly or indirectly influence release and qualification cycles.This research work emphasizes the improvement of test case minimization accuracy by using clustering and classification algorithms. Classification algorithms can predict the class labels of new samples and are further divided into both binary and multiclass algorithms. Logistic regression, developed by David Cox, gives output between 0 and 1 using a sigmoid function but fails to perform well when the data is noisy. On the other hand, the Gaussian Naive Bayes classifier is fast, accurate, and effective in the case of large datasets. It can effectively deal with noise and missing data. Both these models are very important in optimization to make a suitable choice of test cases to enhance the

performance of the system.The Multinomial Naive Bayes classifier is a probabilistic learning algorithm commonly used in Natural Language Processing (NLP). It predicts the label of a text, such as an email or article, by calculating the probability of each possible tag and selecting the one with the highest likelihood. This classifier is particularly effective for discrete features, like word counts in text classification, and can also handle fractional counts such as tf-idf. Meanwhile, the K-means clustering algorithm is an unsupervised learning model designed to group data into k non-overlapping clusters by minimizing the distance between each data point and its corresponding cluster centroid. The algorithm begins by initializing random centroids, assigning data points to the closest centroid based on a distance metric (typically Euclidean distance), and then updating the centroids by recalculating the mean of the data points in each cluster. This process is repeated until cluster assignments stabilize or a set number of iterations is reached. K-means is widely used in various fields for its simplicity and effectiveness in identifying patterns in data.

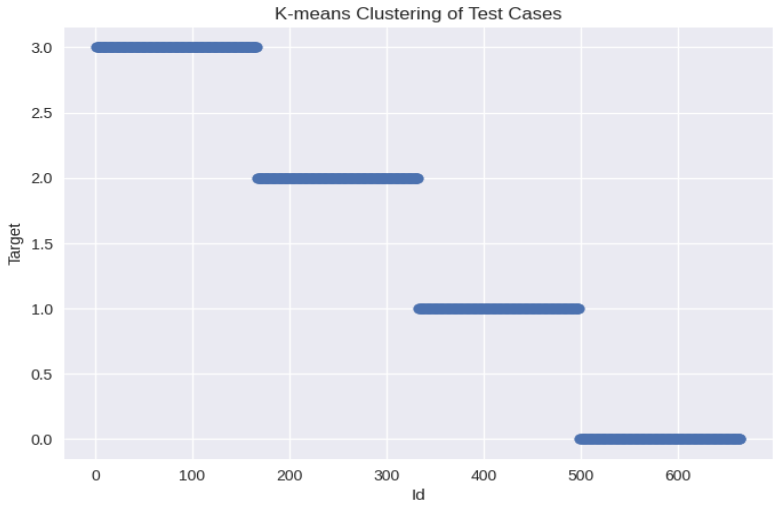

Fig 3: K-means Clustering of Test Cases

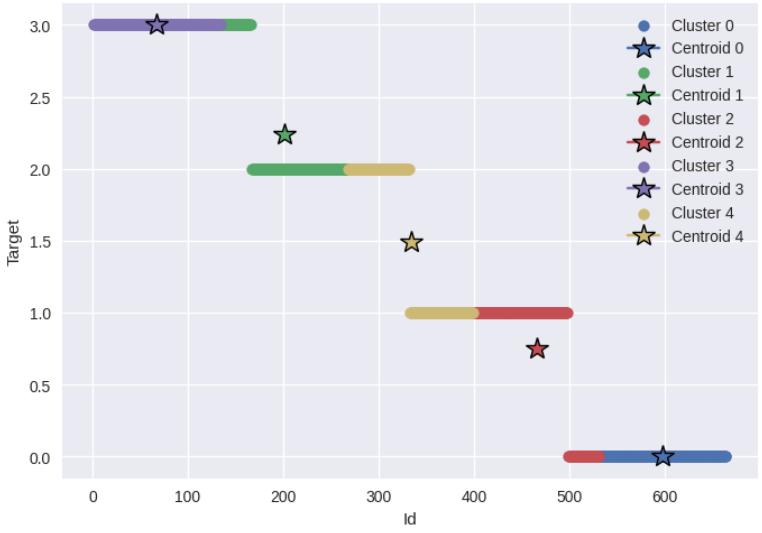

Fig.4: K-means Clustering with Centroid of Test Cases

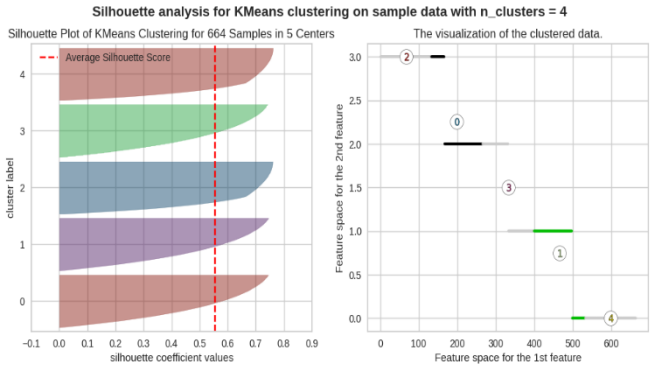

Fig 5: K-means Clustering with Centroid of Test Cases

In a nutshell, the reason for using K-means clustering is to better select and categorize test cases for testing software. In other words, K-means clustering aids in grouping similar test cases through common characteristics, hence optimizing the testing process by lessening redundancy and increasing efficiency. The graph in Fig.3 illustrates an initial clustering of the test cases; data points have been collected together based on their IDs and target labels that point to where the clusters lie. Fig. 4 further expands this visualization by overlaying an asterisk on top of each centroid and using color-coding in the illustration to better show which clusters are separated and at what central point's they are centred. This describes how the test cases are spread in the clusters and how these cluster centres are calculated. Fig. 5 shows the evaluation of clustering performance using silhouette analysis. The silhouette plot on the left side computes cohesion vs separation among all clusters with a red dashed line that indicates the average silhouette score, indicating the general quality of the cluster. To the right, the clustered data is shown in feature space, how clusters of test cases are spatially organized relative to their centroids. These figures together portray the application and effectiveness of K-means clustering in organizing as well as improving the choice of test cases in the testing of software. Table 2 reports accuracy, precision, recall, F1 score, and support for four algorithms: Logistic Regression, Gaussian Naive Bayes, Multinomial Naive Bayes, and K-Means Clustering. These metrics can help to understand the efficiency and reliability of each algorithm with respect to test case classification and clustering.

Table 2: Performance Metrics for Various Algorithms

Algorithm	**Accuracy**	**Precision (Weighted Avg)**	**Recall (Weighted Avg)**	**F1 Score (Weighted Avg)**	**Support**
Logistic Regression	0.84	0.84	0.84	0.84	133
Gaussian Naive Bayes	0.78	0.88	0.78	0.80	133
Multinomial Naive Bayes	0.78	0.88	0.78	0.80	133
K-Means Clustering	1.00	1.00	1.00	1.00	6

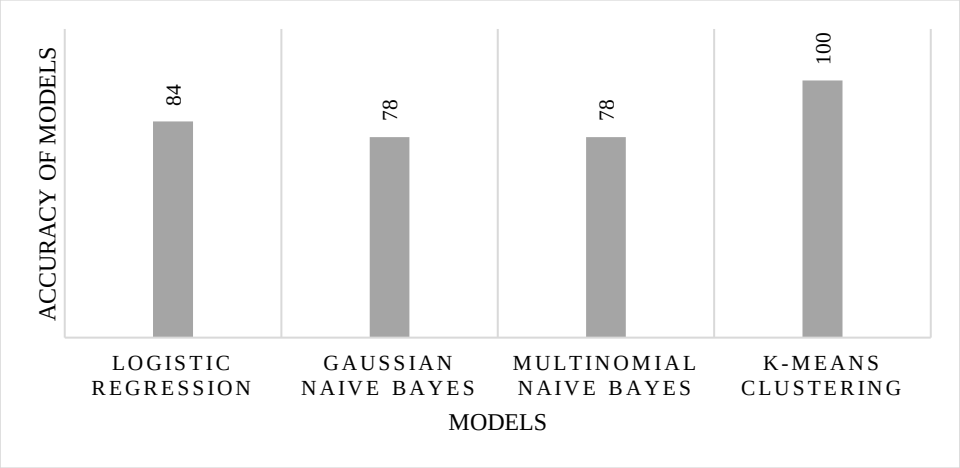

Fig 6: Accuracy Comparison of Classification and Clustering Models

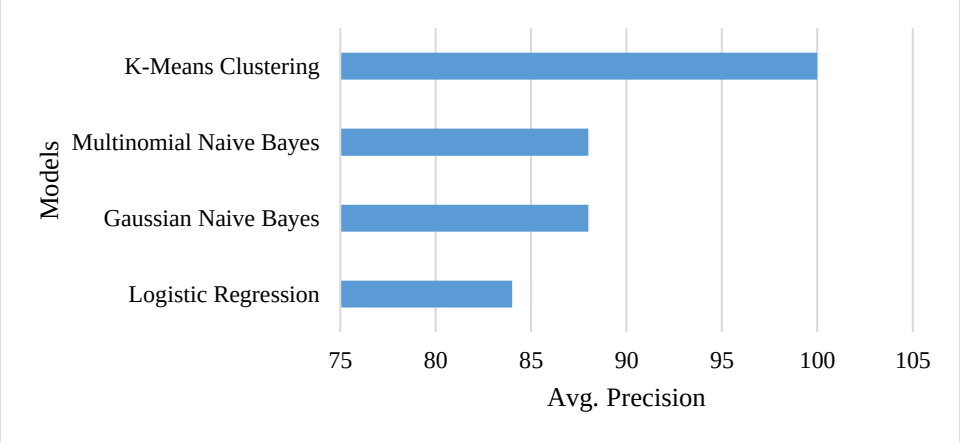

Fig. 7: Comparing Classification and Clustering Models regarding Precision

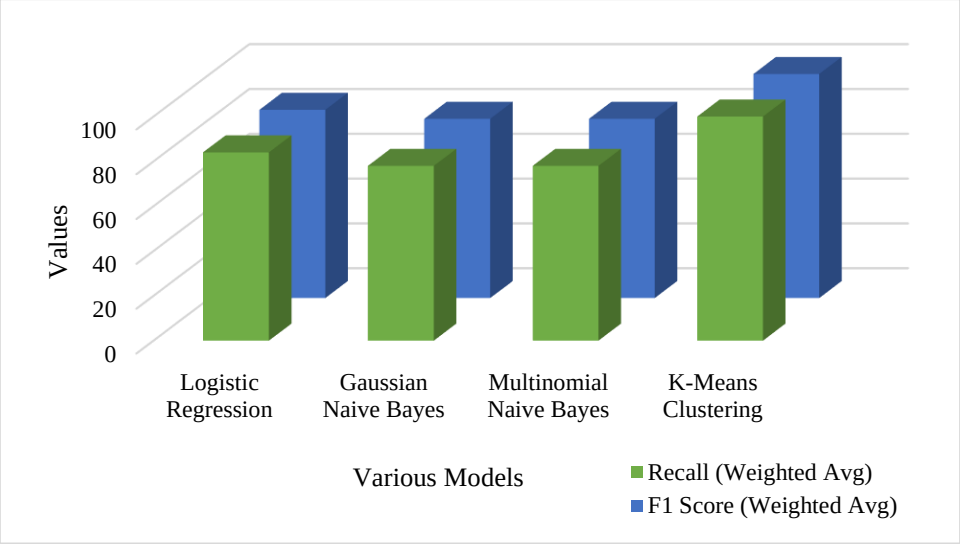

Fig. 8: Comparison of Recall and F1 Score (Weighted Average) Across Different Models

The accuracy of four distinct models—Gaussian Naive Bayes, Multinomial Naive Bayes, K-Means Clustering, and Logistic Regression—is shown in Figure 6. The K-Means Clustering algorithm is the model that performs the best out of all of them, exceeding the others by a significant margin with 100% accuracy. Then comes 84% accurate Logistic Regression, and 78% accurate Gaussian and Multinomial Naive Bayes models. The outstanding results of

K-Means Clustering indicate that it is a useful method for classifying the provided test cases in this circumstance. Four models whose weighted average precision is compared in Fig. 7. With an exceptional precision of 100%, the K-Means Clustering model demonstrates its exceptional capacity to accurately detect real positives while minimizing false positives. About 88% is the precision of the Multinomial Naive Bayes model, and 88% is the precision of the Gaussian Naive Bayes model. With a precision score of approximately 84%, Logistic Regression is the least precise model studied. It appears from this that the K-Means Clustering model is the most accurate for this specific task, while the Naive Bayes models work well but are a little less precise than K-Means.The recall and F1 score (weighted average) of four models are contrasted in Fig. 8. With 100% perfect ratings for both recall and F1 score, K-Means Clustering is the best performer. With recall close to 78% and F1 scores around 80%, Gaussian and Multinomial Naive Bayes perform similarly to Logistic Regression, which likewise performs well with about 84% for both measures. The most successful model overall is K-Means, which is followed by the Naive Bayes and Logistic Regression models.

CONCLUSION

This study introduced and tested a clustering-based approach based on k-means for clustering, with the aim of enhancing the efficiency of the test case minimization which becomes critical in optimized software testing. Most traditional methods have been unable to scale with modern systems, which often are heterogeneous and dynamic, but our approach meets these requirements: by putting together similar test cases into their respective clusters, we managed to remove redundancy whilst ensuring maximum functionality coverage of the software. This approach came with several major benefits: It reduced time consumption for testing, lowered resource usage, saved costs, and provided better detection of faults. Decreased test cases with quality meant reduced execution time and consumption of resources without affecting the quality of the testing process. Moreover, the heterogeneity in the clusters also assured that despite reducing the number of test cases, critical application aspects of the software were tested time and again. All test cases found by the k-means clustering were able to identify most software defects; hence, this method has great application in large-scale projects. Overall, the k-means clustering based test case minimization algorithm gives a very practical and efficient manner to optimize the process of software testing.

REFERENCES

[1] Sinaga, K. P., & Yang, M. S. (2020). Unsupervised K-means clustering algorithm. *IEEE access*, *8*, 80716-80727.

[2] Minh, H. L., Sang-To, T., Wahab, M. A., &Cuong-Le, T. (2022). A new metaheuristic optimization based on K-means clustering algorithm and its application to structural damage identification. *Knowledge-Based Systems*, *251*, 109189.

[3] Chen, J., Zhu, L., Chen, T. Y., Towey, D., Kuo, F. C., Huang, R., & Guo, Y. (2018). Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering. *Journal of Systems and Software*, *135*, 107-125.

[4] Ran, X., Zhou, X., Lei, M., Tepsan, W., & Deng, W. (2021). A novel k-means clustering algorithm with a noise algorithm for capturing urban hotspots. *Applied Sciences*, *11*(23), 11202.

[5] Ismkhan, H. (2018). Ik-means−+: An iterative clustering algorithm based on an enhanced version of the k-means. *Pattern Recognition*, *79*, 402-413.

[6] Syakur, M. A., Khotimah, B. K., Rochman, E. M. S., &Satoto, B. D. (2018, April). Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In *IOP conference series: materials science and engineering* (Vol. 336, p. 012017). IOP Publishing.

[7] Ali, I., Rehman, A. U., Khan, D. M., Khan, Z., Shafiq, M., & Choi, J. G. (2022). Model selection using K-means clustering algorithm for the symmetrical segmentation of remote sensing datasets. *Symmetry*, *14*(6), 1149.

[8] Jain, M., Kaur, G., & Saxena, V. (2022). A K-Means clustering and SVM based hybrid concept drift detection technique for network anomaly detection. *Expert Systems with Applications*, *193*, 116510.

[9] Saputra, D. M., Saputra, D., &Oswari, L. D. (2020, May). Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. In *Sriwijaya international conference on information technology and its applications (SICONIAN 2019)* (pp. 341-346). Atlantis Press.

[10] Fränti, P., &Sieranoja, S. (2019). How much can k-means be improved by using better initialization and repeats?. *Pattern Recognition*, *93*, 95-112.

[11] Vankayalapati, R., Ghutugade, K. B., Vannapuram, R., & Prasanna, B. P. S. (2021). K-Means algorithm for clustering of learners performance levels using machine learning techniques. *Rev. d'IntelligenceArtif.*, *35*(1), 99-104.

[12] Abbas, S. A., Aslam, A., Rehman, A. U., Abbasi, W. A., Arif, S., & Kazmi, S. Z. H. (2020). K-means and k-medoids: Cluster analysis on birth data collected in city Muzaffarabad, Kashmir. *IEEE Access*, *8*, 151847-151855.

[13] Xu, Z., Shen, D., Nie, T., Kou, Y., Yin, N., & Han, X. (2021). A cluster-based oversampling algorithm combining SMOTE and k-means for imbalanced medical data. *Information Sciences*, *572*, 574-589.

[14] Liu, X., Zhu, X., Li, M., Wang, L., Zhu, E., Liu, T., ... & Gao, W. (2019). Multiple kernel $ k $ k-means with incomplete kernels. *IEEE transactions on pattern analysis and machine intelligence*, *42*(5), 1191-1204.

[15] Nguyen, H., Bui, X. N., Tran, Q. H., & Mai, N. L. (2019). A new soft computing model for estimating and controlling blast-produced ground vibration based on hierarchical K-means clustering and cubist algorithms. *Applied Soft Computing*, *77*, 376-386.

[16] Fard, M. M., Thonet, T., &Gaussier, E. (2020). Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters*, *138*, 185-192.

[17] Hussain, S. F., &Haris, M. (2019). A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data. *Expert Systems with Applications*, *118*, 20-34.

[18] Nagy, S. M., Maghawry, H. A., & Badr, N. L. (2023). An Enhanced Approach for Test Suite Reduction Using Clustering and Genetic Algorithms. *Journal of Theoretical and Applied Information Technology (Jatit)*.

[19] Pan, R., Ghaleb, T. A., & Briand, L. (2023, May). Atm: Black-box test case minimization based on test code similarity and evolutionary search. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)* (pp. 1700-1711). IEEE.

[20] Wang, X., & Zhang, S. (2024). Cluster-based adaptive test case prioritization. *Information and Software Technology*, *165*, 107339.

[21] Vo, H. D., & Nguyen, T. T. (2024). CLUE: A CLUSTERING-BASED TEST REDUCTION APPROACH FOR SOFTWARE PRODUCT LINES. *Journal of Computer Science and Cybernetics*, *40*(2), 165-185.

[22] Santhosh Kumar, S. V. N., Palanichamy, Y., Selvi, M., Ganapathy, S., Kannan, A., & Perumal, S. P. (2021). Energy efficient secured K means based unequal fuzzy clustering algorithm for efficient reprogramming in wireless sensor networks. *Wireless Networks*, *27*, 3873-3894.

[23] Yao, Y., Li, Y., Jiang, B., & Chen, H. (2020). Multiple kernel k-means clustering by selecting representative kernels. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(11), 4983-4996.

[24] Moubayed, A., Injadat, M., Shami, A., &Lutfiyya, H. (2020). Student engagement level in an e-learning environment: Clustering using k-means. *American Journal of Distance Education*, *34*(2), 137-156.