Research Article

International Journal of Advanced Computer Research, Vol 12(59)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2021.1152064

K-means based quality prediction of object-oriented software using LR-ACO

Sandeep Ganpat Kamble* **and Animesh Kumar Dubey**
Department of Computer Science, PCST Bhopal, Madhya Pradesh

Received: 11-January-2022; Revised: 25-March-2022; Accepted: 26-March-2022

Abstract
A quality prediction mechanism has been developed in this paper. K-means clustering algorithm has been applied for the clustering of object-oriented features. Finally logistic regression (LR) and ant colony optimization (ACO) (LR-ACO) have been used for the classification. The object-oriented parameters have been considered like polymorphism, encapsulation, abstraction, inheritance and other object-oriented features for experimentation. The purpose of these features to categorize the data in different class levels based on memory usage, reusability and multiple forms. Different hyperparameters like dynamic allocation and feature margin have also been considered for the classification thresholds. Different performance measures have been considered for the experimentation and the results shows the approach effectiveness through different exploration.

Keywords
K-Means, LR, ACO, Polymorphism, Class, Inheritance.

1.Introduction
The current trend in the programming is mainly rely on the proper bug detection as it is crucial in any software monitoring or testing system [1−5]. It also determines the quality of any software. It is important as it will improve the reliability and useful in the progress of the software life cycle. The main components which are used in the quality predictions are data hiding, encapsulation, inheritance and polymorphism. It has been considered to test the dynamic behaviour, usage, configuration assessments, and reusability checking [6−10].

The quality estimation is important due to the need of checking and validating the programming modules [11, 12]. It covers all the tracks, potential software faults at each stage and module synchronization. It can be categorized level wise as it may affect the system mildly or highly. The major problem with the system identification is the data noise, missing values, parameter cohesion and software bugs. It may affect the performance of the system. Different machine learning and approaches can be used for the estimation like K-Nearest Neighbor (KNN), naive Bayes (NB), decision tree (DT), logistic regression (LR), support vector machine (SVM), random forest (RF), linear regression, etc [13−17].

*Author for correspondence

The major challenges in the quality prediction are as follows:
1. Parameter estimation and its association
2. Data arrangement and linking of missing values
3. Feature extraction and categorization
4. Object oriented features contribution

Considering the above challenges, the main objective of this paper is to cluster object-oriented features considering different class labels and apply classification for the parametric analysis based on feature optimization. For this reason, k-means clustering algorithm has been applied for the clustering of object-oriented features and then LR and ant colony optimization (ACO) (LR-ACO) have been used for the classification.

This paper is organized as follows. Review of literature have been explored in section 2. Method has been discussed in section 3. Section 4 investigated the result and elaborated the discussion. Section 5 covers the summary with concluding remarks.

2.Literature review
This section discusses the related work in the domain of object-oriented programming defects. In 2017, Hosseini et al. [18] carried out a methodical literature review. To address research concerns, primary study

findings are combined (thematic, meta-analysis). They discovered 30 primary studies that met quality standards. The choice of metrics affects performance measures, with the exception of precision. The most typical measurements are recall, precision, f-measure, and AUC. In cross project defect prediction (CPDP), models based on decision trees and nearest neighbours typically perform better than models based on the widely used NB. In 2018, Zaffar et al. [19] concentrated on the crucial characteristics of students taking programming classes that were employed in some of the earlier research. They also highlighted various classification prediction techniques to forecast students' achievement in programming courses. They took action to raise educational standards, and all those involved in education benefited from their efforts. In 2019, Glazier and Garlan [20] suggested a method that allowed the development of a meta-manager, a higher-level autonomic system that neither directly coordinates the operations of the sub-autonomic managers nor subsumes the control functions. Instead, they packed and abstract each subsystem's behaviour into a parameterized adaptation policy that the meta-manager can modify to fine-tune the subsystem adaption's adaptive behaviour. Further, they provided a method for meta-managing a set of autonomous subsystems that preserves local autonomy while allowing for the synthesis of stochastic game strategies to increase global aggregate utility. In 2019, KS [21] forecast software reliability by modelling the relationship between object-oriented design metrics and object-oriented programme dependability. The system is made up of modules for the classifier, testing, and reliability computation. The system is made up of classifier, metrics creation, testing, and reliability calculation modules. In 2020, Miholca and Oneţ-marian [22] offered three software metrics suites derived from conceptual and structural connection, and they examined how various combinations of these metrics affect how well software defect prediction model's function. They evaluated the relative performance of the models while evaluating the conceptual connection utilising features extracted with LSI versus Doc2Vec in combination with Cosine versus Euclidean similarity. The datasets used were ant 1.7, jEdit 3.2, 4.0, 4.1, 4.2, 4.3 and tomcat 6.0 and the method used were first, COMET and Promise. The applied method logistic regression, KNN and artificial neural network were applied on the used dataset. The logistic regression achieved the highest value of 0.917 in first method of jEdit 4.3 dataset whereas KNN achieved the highest value of 0.948 in COMET method of jEdit 4.3 and ANN achieved the highest value of 0.919 in first method of jEdit 3.2 dataset. In 2021, Azzeh et al. [23] performed review to categories, use case points (UCP) effort estimation studies according to four factors: contribution kind, research methodology, dataset type, and UCP methodologies. To examine these works from a variety of angles, including estimate accuracy, a supportive estimation environment, and the effect of combining approaches on UCP accuracy. The UCP research' main methodologies were enhancement and construction procedures. The development of estimate tools, particularly intelligent systems that convert use case descriptions to UCP metrics, received little attention. Additionally, they looked into a few journals to assess the range of prospective sources for estimation studies and awareness of published results. In 2021, Jin [24] suggested a novel distance metric learning approach that relies on cost sensitive learning to lessen the effect of sample class imbalance. This approach is then applied to the large margin distribution machine (LDM) to replace the conventional kernel function. Further, they proposed the improved cost sensitive (CS) LDM. The improved CSLDM (CS-ILDM) was applied to five publicly available datasets gathered from NASA metric data repository. In addition to having acceptable predictive performance, CS-ILDM also has the lowest cost per incorrect prediction. In 2021, Colakoglu et al. [25] examined the current topics of study and developments on this subject that have appeared in the literature during the past ten years. On 70 articles and conference papers that were published between 2009 and 2019 on software product quality metrics (SPQM) as indicated in their titles and abstracts, a systematic mapping analysis was conducted. The outcome is presented via diagrams, explanatory text, and the mind mapping technique. The results contain a trend map from 2009 to 2019, information about this field and measuring tools, concerns identified as having room for advancement, and consistency between conference papers, journals, and internationally recognized quality models. In 2021, Feng et al. [26] used the stable Synthetic Minority Oversampling Technique (SMOTE) to reduce the randomness. They applied the DT, KNN, RF and SVM as the four classifiers on 26 publicly available datasets from PROMISE repository. In terms of AUC, balance, and MCC, respectively, the difference between the worst and greatest performances of SMOTE-based oversampling approaches can reach up to 23.3%, 32.6%, and 204.2%. In 2021, Matloob et al. [27] offered concise details on the most recent trends and

developments in ensemble learning for predicting software defects and established a foundation for further advancements and additional reviews. They discovered that the random forest, boosting, and bagging ensemble approaches are often used by researchers. The use of stacking, voting, and Extra Trees is less common. They discovered that many studies have suggested a number of interesting frameworks that use ensemble learning techniques, including EMKCA, SMOTE-Ensemble, MKEL, SDAEsTSE, TLEL, and LRCR. In 2021, Meng et al. [28] suggested a semi-supervised software defect prediction model which combined the feature normalisation, over-sampling technology, and a Tri-training process. In order to remove the impact of excessively high or excessively low feature values on the model's classification performance, the feature data are first smoothed using the feature normalisation approach. Second, the uneven categorization of labelled samples is resolved by expanding and sampling the data using the oversampling method. The Tri-training approach then creates a fault prediction model using machine learning on the training samples. They used the CM1, PC1, KC1 and KC2 as the dataset from Promise library. They used the Naïve Bayes, Decision Tree, AdaBoost, S4VM+ and Tri_SSDPM as the classifiers. The accuracy of 0.874, precision of 0.886, recall of 0.828 and F-measure of 0.855 was obtained. In 2021 Ming et al. [29] put forth the dynamic anchor learning (DAL) approach, which makes use of the newly established matching degree to thoroughly assess the anchors' capability for localization and performs a quicker label assignment process. With just a few horizontal preset anchors, they can now recognise arbitrary-oriented objects more effectively because of the newly added DAL. Results from experiments on the three remote sensing datasets HRSC2016, DOTA, and UCAS-AOD as well as the scene text dataset ICDAR 2015 demonstrated that their strategy significantly outperformed the baseline model. They achieved the mAP/F score of 81.3, 88.3 and 76.1 in ICDAR 2013, NWPU VHR-10 and VOC 2007. In 2021, Mustaqeem and Saqib [30] proposed a hybrid machine learning strategy combining Principal component analysis (PCA) and SVM. In order to carry out their investigation, they used PROMISE dataset of 344 observations of CM1, 2109 observations of KC1 dataset from the NASA directory. The dataset was then divided into training and testing datasets of 104 observation from CM1 dataset, and 633 observations from KC1 dataset. Additionally, they used the GridSearchCV technique to fine-tune the hyperparameters. They discovered that the proposed hybrid model has higher accuracy of 95.2% and 86.6% in CM1 and KC1 datasets. In Serban et al. [31] used the Landsat library of satellite pictures from 1973 to 2019 to evaluate the LUCC. Combining the Support Vector Machine (SVM) technique with object-based image analysis (OBIA), a theme change detection analysis was carried out. For the years 1973, 1986, 2000, and 2019, four types of LUCC (forest, grass, water, and anthropic) were retrieved with an overall accuracy of >90%. In 2021, Sun et al. [32] collaborative filtering-based source projects selection (CFPS) method, which is based on collaborative filtering. They applied C4.5, LR, NB, RF and sequential minimal optimization as the machine learning algorithms. The average MAP of 0.6518 and average MRR of 0.8688 was received. In 2021, Tahir et al. [33] used a regression-based approach. They were able to determine the influence based on the defects. In 2021, Wu et al. [34] proposed the multi-source heterogeneous cross-project defect prediction (MHCPDP) technique. It has been used for the detection of unrelated features. In 2022, Gu et al. [35] established an object-oriented probability integration prediction model framework. They introduced an object-oriented method coupled with the traditional probability integration method and then created the framework. The primary variables influencing mining subsidence, namely the subsidence coefficient, thickness, and dip angle, were subjected to sensitivity analysis. Engineering scenarios are used to test the predictability of the model, and the primary mining subsidence impacting elements are also examined. The outcomes demonstrated that the projected outcomes of the model are essentially compatible with the situation. In 2022, Guo et al. [36] combined an object-oriented methodology and a random forest algorithm to provide an effective and practical method for identifying urban trees. Finally, utilising the RF, SVM, and KNN classifiers, the categorization of urban trees was carried out based on the nine methods. The findings demonstrate that the RF classifier outperforms SVM and KNN.

The analysis of some of the selected paper is shown in *Table 1*.

Table 1 Analysis of the selected papers

S. No.	Author	Dataset	Method	Approach	Result	Limitation
1	Aktaş and Buzluca (2018) [37]	5,973 Mylyn task sessions from Eclipse Bugzilla were used in total.	They identified the most pertinent subset of measurements, they employed the PCA and correlation-based feature selection (CFS) methodologies. Then lastly, they applied Random Forest to find error prone classes.	They suggested a method for creating machine learning-based models for predicting bugs. They address issues with labelling and feature selection.	Values for recall, precision, and f-measure are 0.725, 0.828, and 0.773, respectively.	Each project has its own personality, hence multiple models should be developed for various software systems.
2	Chhiba et al. (2018) [38]	9 and 7 java software development projects.	They created the "Maintainability Estimation Model," a multivariate linear model that assesses the maintainability in terms of its self-description, simplicity, and modularity.	According to their self-description, simplicity, and modularity, the maintainability is estimated by the maintainability model. While R.C. Martin's measures are used to measure simplicity and modularity,	R^2 achieved for modularity model is 0.862 and for similarity model is 0.958	More advanced maintainability assessment models can be created by conducting a broader study with a range of industrial projects in various fields.
3	Tripathi et al. (2018) [39]	QWS Dataset	They suggested creating prediction models based on 16 aggregate measures and demonstrated that these aggregate measures differed significantly from one another. Six feature selection methods are used to choose the optimum feature subset, and they used Extreme Learning Machines (ELM) with various kernels to create prediction models.	By utilising CK and object-oriented metrics of the underlying Java files, they carried out an empirical investigation on the prediction of 12 QoS factors for online services. For each WSDL, they combined CK and OO scores using 16 different measurements. To identify important characteristics, six feature ranking and extraction algorithms are applied.	The ELM with RBF kernel has a mean and median MMRE of 0.369 and 0.346, respectively.	-------
4	Yang et al. (2018) [40]	Satellite remote sensing ship image dataset (SRSS)	They proposed an innovative multitask rotational area convolutional neural network-based detection approach.	They used the rotating bounding box regression, adaptive region of interest (ROI) align, prow direction prediction, and rotational non maximum suppression make up the majority of this model (R-NMS).	They achieved recall of 85.2%, precision of 84.5% and F1-score of 84.9%.	Objects with significant levels of overlap are constrained.
5	Ha et al. (2019) [41]	They used JM1, PC4, KC2, MC1, KC1, PC3, CM1, MW1, PC1, Class, MC2, KC3 and PC2 datasets from PROMISE repository.	They applied LR, KNN, DT, RF, SVM and multilayer perceptron	For defect prediction, they used object-oriented metrics and other machine learning approaches. The PROMISE datasets were used for their experimental investigation, which focused on seven well-known machine learning methods for defect prediction. According to the results, Multilayer Perceptron performs best for method-level datasets, whereas Support Vector Machine	The highest F1 score, AUC and accuracy of 0.59, 0.91 and 0.91 was achieved in Multilayer Perceptron.	There is a need to study classification approaches to address the problem of dataset imbalance in fault prediction.

S. No.	Author	Dataset	Method	Approach	Result	Limitation
				works best for class-level datasets.		
6	Kabir et al. (2019) [42]	jm1 and prop defected datasets.	They utilised the drift detection method (DDM) strategy and used the chi-square test with Yates continuity adjustment to assess its statistical significance. They applied Naïve Bayes and Decision Tree as the classifiers.	They used the widely acceptable and well-known DDM drift detection approach to look for concept drift in the jm1 and prop datasets.	The p-value received for 3000 historical datasets from NB and DT are 0.027 and 0.025.	It is necessary to run an experiment using Fishers Exact test on small datasets of software defects to track changes in data distributions that affect the accuracy of the predictions.
7	Zhang et al. (2019) [43]	Two publicly accessible datasets, such as Emotions and Scene, are modified, along with one real-world Herbs dataset.	For the multi-modal and multi-label (MMML) problem, they introduced a novel instance-oriented multi-modal classifier chains (MCC) technique that can produce convincing predictions with partial modalities.	They suggested an MCC model for the MMML problem that is inspired by adaptive decision approaches, taking into account both multi-label learning and feature extraction.	They obtained MCC of 2.520±0.182 for Herbs, 1.519±0.313 for emotions and 2.109±0.324 for scene dataset.	It is time consuming.
8	Afric et al. (2020) [44]	They used the PROMISE dataset repository's CM1, JM1, KC1, KC2, and PC1 datasets.	They applied reconstruction error probability distribution (REPD) model, which can deal with individual and group abnormalities. Further, they compared it to five models: decision tree, logistic regression, k-nearest neighbours, and hybrid SMOTE-Ensemble on five different traditional code feature datasets.	For the purpose of predicting source code defects, they presented the REPD model. They used the autoencoder neural network architecture, which showed promising results in anomaly detection tasks, to identify flaws in software artefacts.	They demonstrate that their model generates noticeably superior outcomes, raising the F1-score by up to 7.12%.	It is necessary to look at other alternative anomaly detection methods that can be useful for the subject of predicting source code defects.
9	Balogun et al. (2021) [45]	Different datasets selected from NASA were like: CM1, KC1, KC2, KC3, MW1, PC1, PC3, PC4 and PC5.	They used DT and NB models.	It has been used for software fault prediction.	The average accuracy achieved from Naïve Bayes and Decision Trees was 76.33 and 83.01.	It is worthwhile to investigate the impact of threshold values on the efficacies of FFS.
10	Khurma et al. (2021) [46]	21 publically available datasets were used.	They proposed a binary variant of Moth flame optimization (MFO) algorithm by using Island BMFO. Further, it is compared by SVM, NB and KNN.	In order to improve the BMFO and address the feature selection issue in the context of software defect prediction.	With an average G-mean of 78%, IsBMFO followed by SVM classification is the best model for the SDP problem when compared to other models.	There is need of island-based investigation.

3.Methods

In this paper, k-means clustering was used for the object-oriented clustering features and then LR and ACO have been used for the classification. The phases are as follows:

1. Data pre-processing: For the experimentation object-oriented programming have been considered. We have considered the code of C++ and java programming modules.
2. Data clustering**:** K-means clustering was used for the object-oriented clustering features. For the comparison of the observed and expected results, chi-square analysis has been used.

K-means algorithm

Step 1: Initialize the number of clusters.

Step 2: Assign, K random points. Categorize according to the data point.

Step 3: Centroid computation.

Step 4: The above steps will be iterated till the variation is minimized or nil.
4.1 Calculate the Euclidean distance for calculating the distance between the data points.
4.2 Allocate the data point to the nearest centroid.
4.3 Finalize the data points.

3. Classification:Combination of LR and ACO algorithms have been used for the classification.

Logistic regression:
Step 1: Assignments of linear weight. Parameter initialization has been performed in this step.
Step 2: Predict the probability value based on independent features for the dependent variable.
Step 3: Error margin calculated based on the predicted value.
Step 4: Final estimation on the same region.
Step 5: Split the data for training and testing phase.
Step 6: Final estimated value

ACO Algorithm:
Step 1: Population initialization (Input the classified LR values).
Step 2: Based on the categorized faults, initialize the pheromone values.
Step 3: The above step has been repeated till the weight assignment for each factor.
Step 4: Construct the optimized solution.
Step 5: Update the complete trails and stop the iterations.
The working process is depicted clearly in Figure 1.

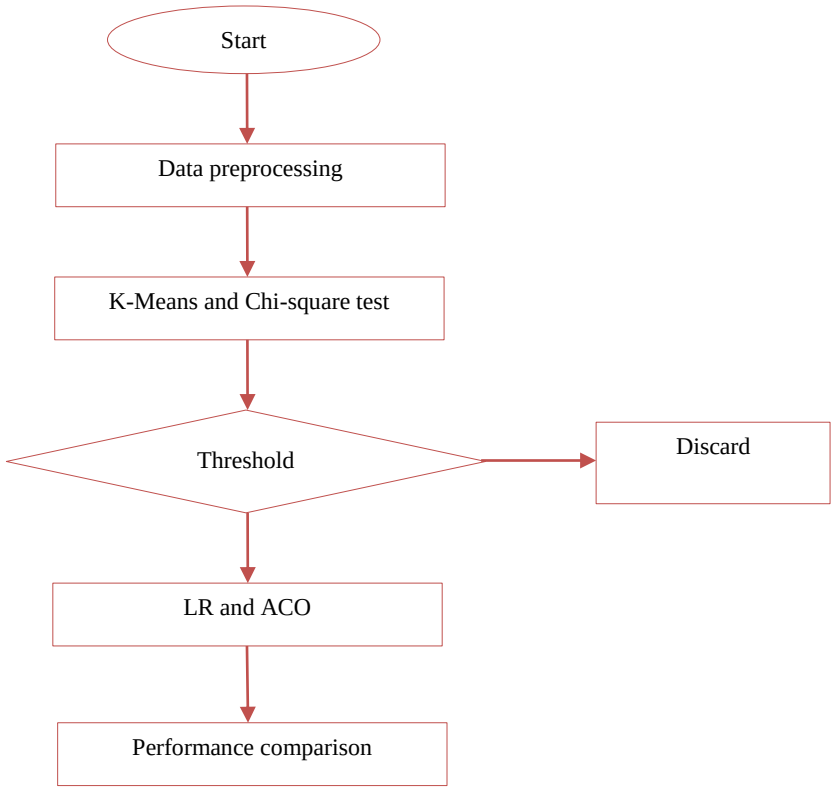

Figure 1 Working process of the complete approach

4. Results and discussion

In this section results have been discussed and investigated. The performance metrics considered here are F-score (FS), ratio (defect odd) (Rd) and deviation margin (DM). It has been calculated as follows:

$$FS = (2 \times P \times R) / (P+R) \quad (1)$$

$$Rd = 2 \times R (1-P) / (1-P \times R) \quad (2)$$

$$DM = ((1-P) \times k-(1-R) \times k) \quad (3)$$

Here precision is denoted by P, recall is indicated by R. k value is 2.

The result for FS based on object-oriented parameters like class module, object module, inheritance module and polymorphism along with the Rd and deviation margin have been discussed here. *Figure 2* shows the comparison of FS value considering different object-oriented modules (set-1). *Figure 3* shows the

comparison of FS value considering different object-oriented modules (set-2). *Figure 4* shows the comparison of FS value considering different object-oriented modules (set-3). *Figure 5* shows the comparison of Rd value considering different object-oriented modules (set-1). *Figure 6* shows the comparison of Rd value considering different object-oriented modules (set-2). *Figure 7* shows the comparison of Rd value considering different object-oriented modules (set-3). *Figure 8* shows the comparison of deviation margin considering different object-oriented modules (set-1). *Figure 9* shows the comparison of deviation margin considering different object-oriented modules (set-2). *Figure 10* shows the comparison of deviation margin considering different object-oriented modules (set-3). The results shows that the performance measures are capable in the detection of deviation margin in different object-oriented modules. The size of the modules is varied between 10 KB to 10 MB.

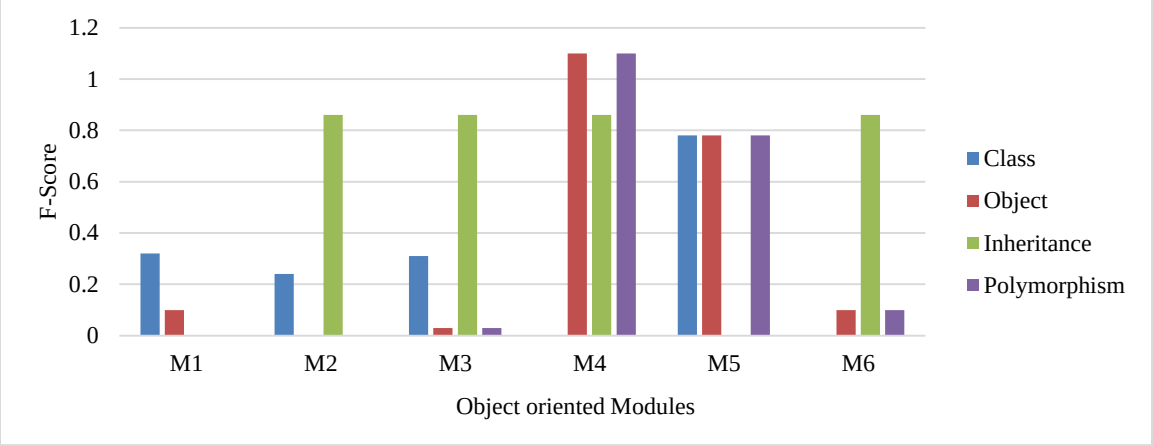

Figure 2 Comparison of FS value considering different object-oriented modules (set-1)

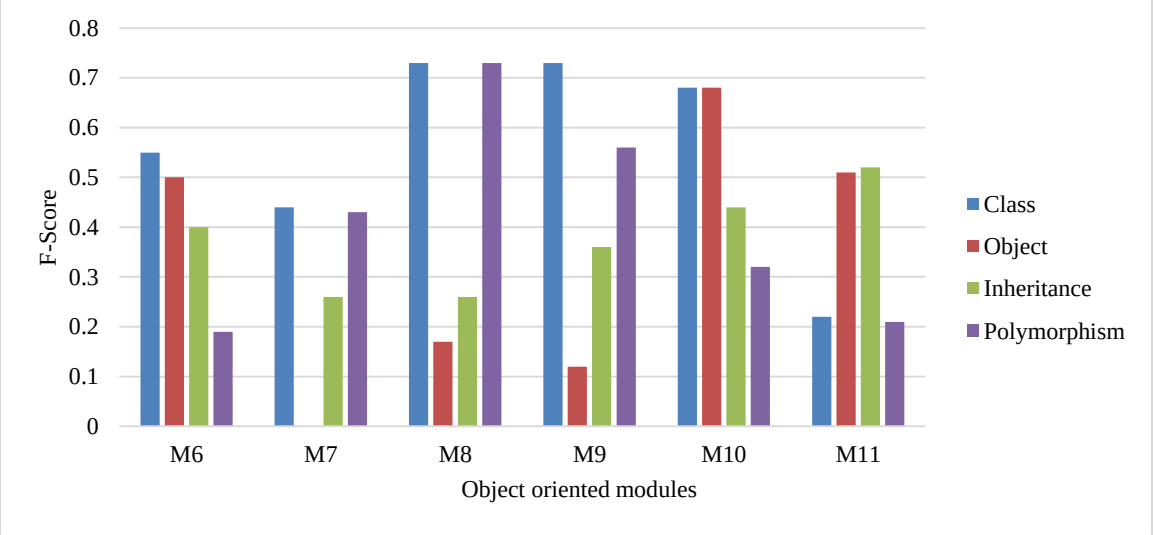

Figure 3 Comparison of FS value considering different object-oriented modules (set-2)

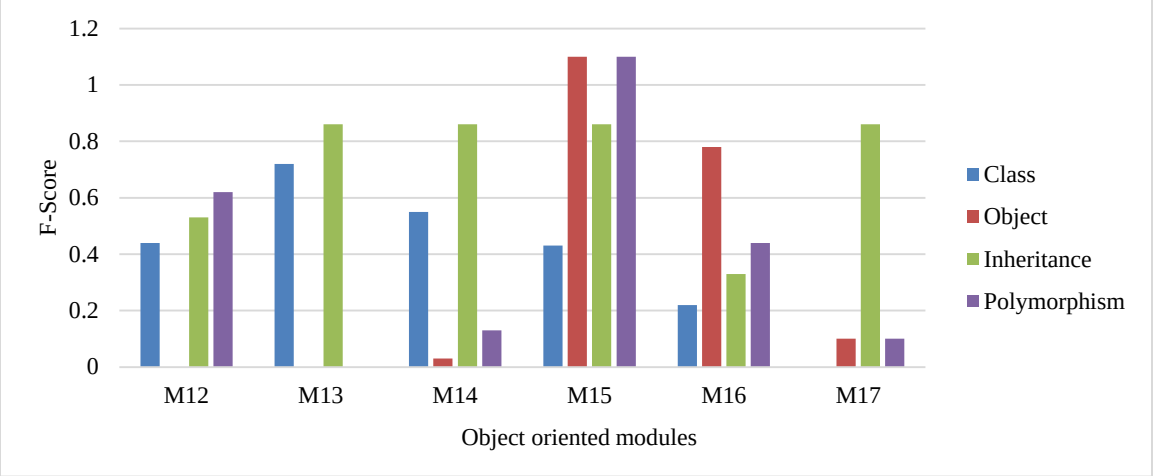

Figure 4 Comparison of FS value considering different object-oriented modules (set-3)

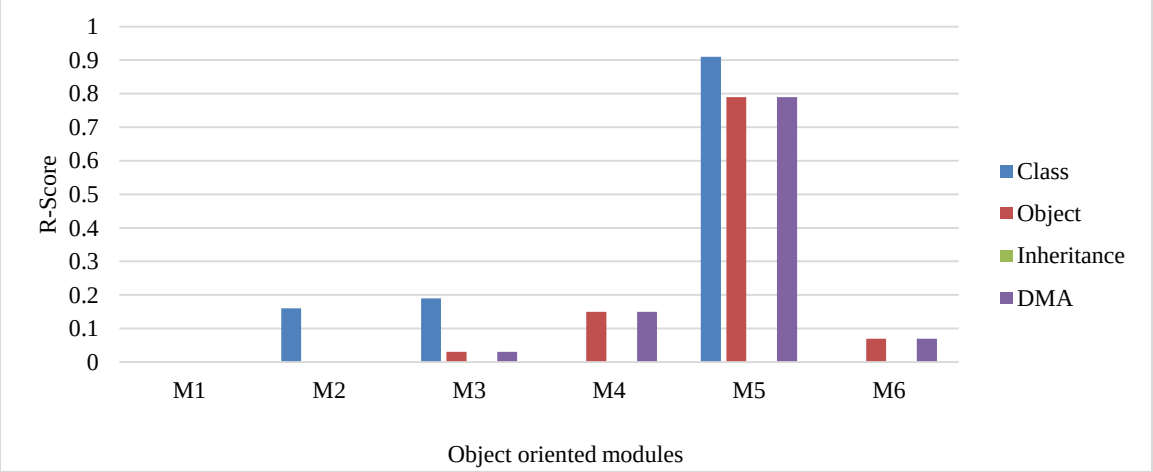

Figure 5 Comparison of Rd value considering different object-oriented modules (set-1)

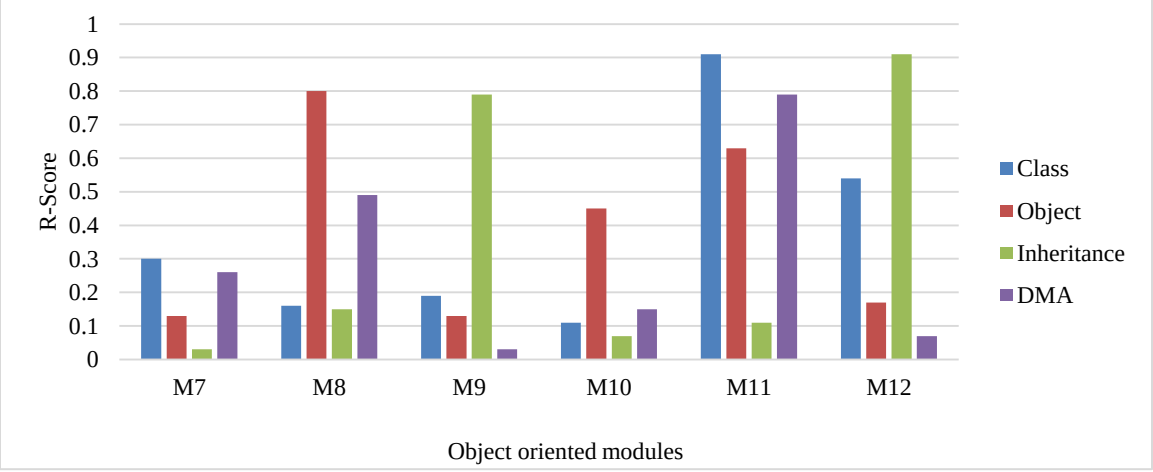

Figure 6 Comparison of Rd value considering different object-oriented modules (set-2)

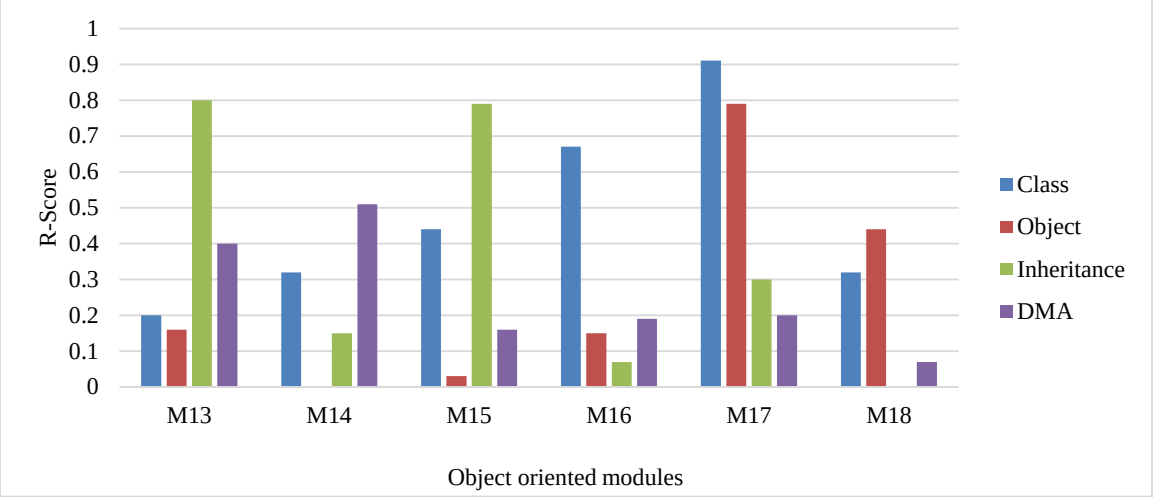

Figure 7 Comparison of Rd value considering different object-oriented modules (set-3)

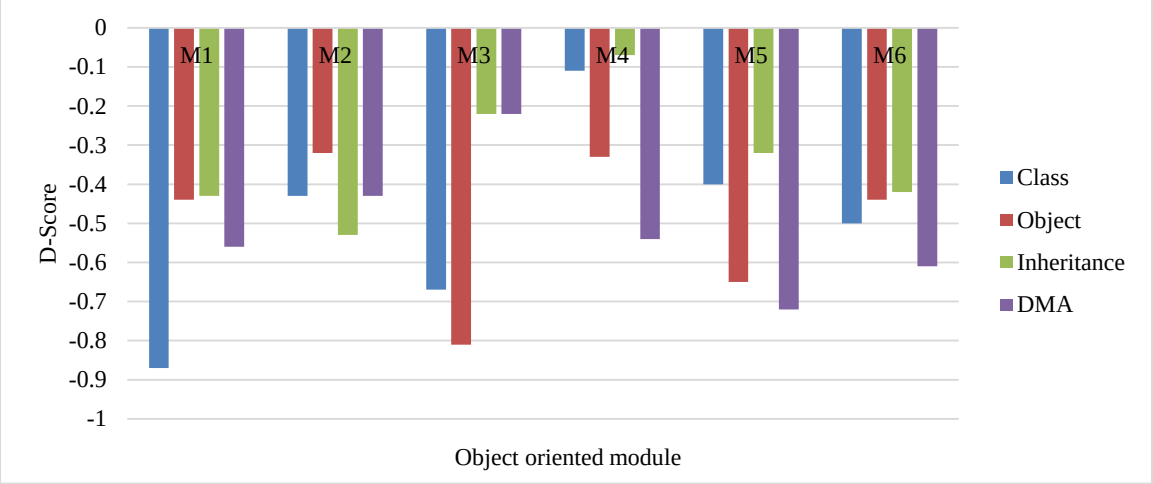

Figure 8 Comparison of deviation margin considering different object-oriented modules (set-1)

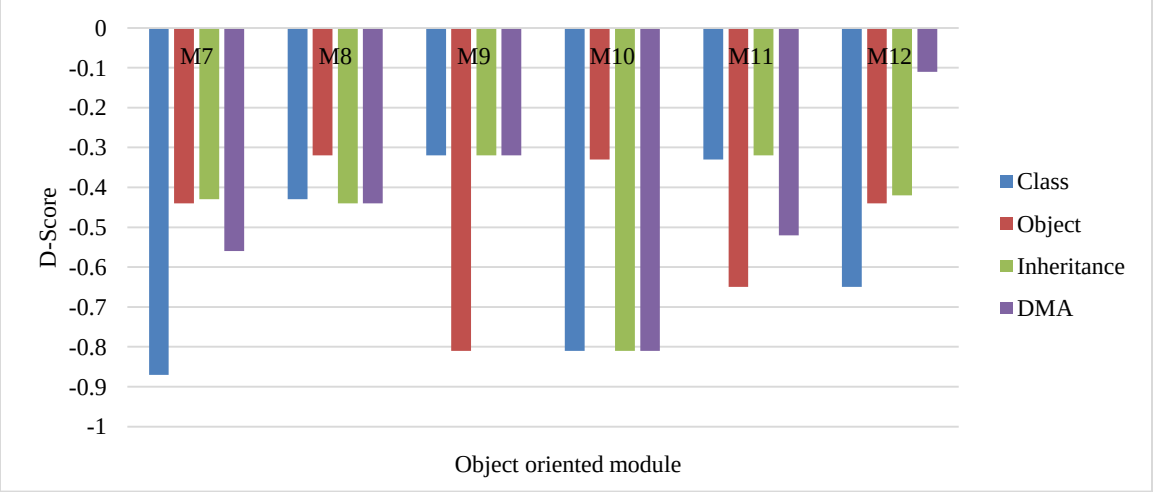

Figure 9 Comparison of deviation margin considering different object-oriented modules (set-2)

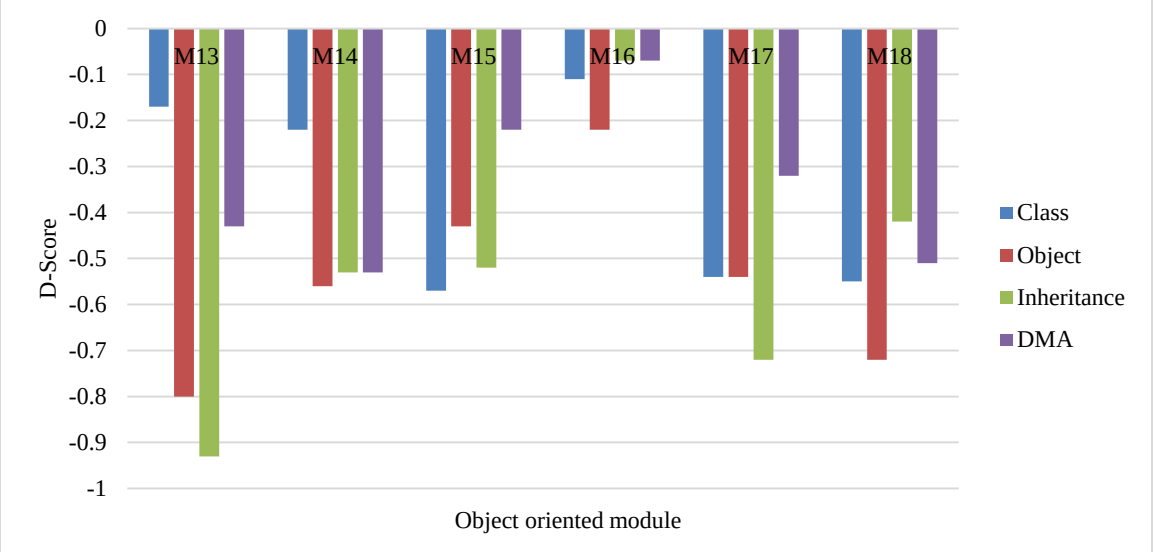

Figure 10 Comparison of deviation margin considering different object-oriented modules (set-3)

5. Conclusion

In this paper, k-means clustering algorithm has been applied for the clustering of object-oriented features along LR and ACO for the classification. Data pre-processing, data clustering and classification have been performed for the software quality estimation considering object-oriented parameters. The performance metrics considered are FS, Rd and DM. The size of the modules is random and variable. The modules belong to C++ and java codes considering object-oriented parameters. The results indicate that the approach is capable in calculating the performance measure with less error margin.

Acknowledgment

None.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Jorayeva M, Akbulut A, Catal C, Mishra A. Machine learning-based software defect prediction for mobile applications: a systematic literature review. Sensors. 2022; 22(7):1-17.

[2] Yadav DK, Azad C, Singh J, Adhikary DR. CIAFP: a change impact analysis with fault prediction for object-oriented software. International Journal of Software Innovation (IJSI). 2022; 10(1):1-9.

[3] Saha JK, Patidar K, Kushwah R, Saxena G. Object oriented quality prediction through artificial intelligence and machine learning: a survey. ACCENTS Transactions on Information Security. 2020; 5 (17): 1-5.

[4] Kumar P, Singh SN, Dawra S. Software component reusability prediction using extra tree classifier and enhanced Harris hawks optimization algorithm. International Journal of System Assurance Engineering and Management. 2022; 13(2):892-903.

[5] Dubey AK, Kushwaha GR, Shrivastava N. Heterogeneous data mining environment based on dam for mobile computing environments. In international conference on advances in information technology and mobile communication 2011 (pp. 144-9). Springer, Berlin, Heidelberg.

[6] Goyal S. Handling class-imbalance with KNN (neighbourhood) under-sampling for software defect prediction. Artificial Intelligence Review. 2022; 55(3):2023-64.

[7] Pandit M, Gupta D, Anand D, Goyal N, Aljahdali HM, Mansilla AO, et al. Towards design and feasibility analysis of DePaaS: AI based global unified software defect prediction framework. Applied Sciences. 2022; 12(1):1-26.

[8] Goyal S. Effective software defect prediction using support vector machines (SVMs). International Journal of System Assurance Engineering and Management. 2022; 13(2):681-96.

[9] Cabral GG, Minku LL. Towards reliable online just-in-time software defect prediction. IEEE Transactions on Software Engineering. 2022.

[10] Dubey AK, Gupta U, Jain S. Computational measure of cancer using data mining and optimization. In international conference on sustainable communication networks and application 2019 (pp. 626-32). Springer, Cham.

[11] Pemmada SK, Behera HS, Nayak J, Naik B. Correlation-based modified long short-term memory network approach for software defect prediction. Evolving Systems. 2022:1-9.

[12] Jain B, Patidar S, Sudershan D. Heterogeneous software defect prediction using generative models. In 11th international conference on communication

systems and network technologies (CSNT) 2022 (pp. 367-72). IEEE.
[13] Jena SD, Kaur J, Rani R. A review of prediction of software defect by using machine learning algorithms. Recent Innovations in Computing. 2022:61-70.
[14] Jain S. Medical data clustering and classification using TLBO and machine learning algorithms. Computers, Materials and Continua. 2021; 70(3):4523-43.
[15] Goyal J, Ranjan Sinha R. Software defect-based prediction using logistic regression: review and challenges. In second international conference on sustainable technologies for computational intelligence 2022 (pp. 233-48). Springer, Singapore.
[16] Rajnish K, Bhattacharjee V, Gupta M. A novel convolutional neural network model to predict software defects. Fundamentals and Methods of Machine and Deep Learning: Algorithms, Tools and Applications. 2022: 211-35.
[17] Dubey AK, Kapoor D, Kashyap V. A review on performance analysis of data mining methods in IoT. International Journal of Advanced Technology and Engineering Exploration. 2020; 7(73):193-200.
[18] Hosseini S, Turhan B, Gunarathna D. A systematic literature review and meta-analysis on cross project defect prediction. IEEE Transactions on Software Engineering. 2017; 45(2):111-47.
[19] Zaffar M, Hashmani MA, Savita KS. A study of prediction models for students enrolled in programming subjects. In 2018 4th international conference on computer and information sciences (ICCOINS) 2018 (pp. 1-5). IEEE.
[20] Glazier T, Garlan D. An automated approach to management of a collection of autonomic systems. In IEEE 4th international workshops on foundations and applications of self* systems (FAS* W) 2019 (pp. 110-5). IEEE.
[21] KS VK. A method for predicting software reliability using object oriented design metrics. In international conference on intelligent computing and control systems (ICCS) 2019 (pp. 679-82). IEEE.
[22] Miholca DL, Oneț-Marian Z. An analysis of aggregated coupling's suitability for software defect prediction. In 22nd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC) 2020 (pp. 141-8). IEEE.
[23] Azzeh M, Nassif AB, Attili IB. Predicting software effort from use case points: a systematic review. Science of Computer Programming. 2021; 204:1-26.
[24] Jin C. Software defect prediction model based on distance metric learning. Soft Computing. 2021; 25(1):447-61.
[25] Colakoglu FN, Yazici A, Mishra A. Software product quality metrics: a systematic mapping study. IEEE Access. 2021; 9:44647-70.
[26] Feng S, Keung J, Yu X, Xiao Y, Zhang M. Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction. Information and Software Technology. 2021; 139:106662.
[27] Matloob F, Ghazal TM, Taleb N, Aftab S, Ahmad M, Khan MA, et al. Software defect prediction using ensemble learning: a systematic literature review. IEEE Access. 2021:1-8.
[28] Meng F, Cheng W, Wang J. Semi-supervised software defect prediction model based on tri-training. KSII Transactions on Internet and Information Systems (TIIS). 2021; 15(11):4028-42.
[29] Ming Q, Zhou Z, Miao L, Zhang H, Li L. Dynamic anchor learning for arbitrary-oriented object detection. In proceedings of the AAAI conference on artificial intelligence 2021 (pp. 2355-63).
[30] Mustaqeem M, Saqib M. Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection. Cluster Computing. 2021; 24(3):2581-95.
[31] Șerban RD, Șerban M, He R, Jin H, Li Y, Li X, et al. 46-year (1973–2019) permafrost landscape changes in the hola basin, northeast china using machine learning and object-oriented classification. Remote Sensing. 2021; 13(10):1-19.
[32] Sun Z, Li J, Sun H, He L. CFPS: collaborative filtering based source projects selection for cross-project defect prediction. Applied Soft Computing. 2021; 99:1-13.
[33] Tahir A, Bennin KE, Xiao X, MacDonell SG. Does class size matter? an in-depth assessment of the effect of class size in software defect prediction. Empirical Software Engineering. 2021; 26(5):1-38.
[34] Wu J, Wu Y, Niu N, Zhou M. MHCPDP: multi-source heterogeneous cross-project defect prediction via multi-source transfer learning and autoencoder. Software Quality Journal. 2021; 29(2):405-30.
[35] Gu Z, Zhao Y, Gao R, Wu L. Research on the prediction model of mine subsidence based on object-oriented and probability integration method. Geofluids. 2022; 2022:1-14.
[36] Guo Q, Zhang J, Guo S, Ye Z, Deng H, Hou X, et al. Urban tree classification based on object-oriented approach and random forest algorithm using unmanned aerial vehicle (UAV) multispectral imagery. Remote Sensing. 2022; 14(16):1-17.
[37] Aktaş F, Buzluca F. A Learning-based bug predicition method for object-oriented systems. In IEEE/ACIS 17th international conference on computer and information science (ICIS) 2018 (pp. 217-23). IEEE.
[38] Chhiba L, Abdelouahid RA, Marzak A. Predicting maintainability of object-oriented system. In international conference on control, automation and diagnosis (ICCAD) 2018 (pp. 1-5). IEEE.
[39] Tripathi MK, Chaubisa D, Kumar L, Neti LB. Prediction of quality of service parameters using aggregate software metrics and machine learning techniques. In 15th IEEE India council international conference (INDICON) 2018 (pp. 1-6). IEEE.
[40] Yang X, Sun H, Sun X, Yan M, Guo Z, Fu K. Position detection and direction prediction for arbitrary-oriented ships via multitask rotation region convolutional neural network. IEEE Access. 2018; 6:50839-49.

[41] Ha TM, Tran DH, Hanh LT, Binh NT. Experimental study on software fault prediction using machine learning model. In 2019 11th international conference on knowledge and systems engineering (KSE) 2019 (pp. 1-5). IEEE.
[42] Kabir MA, Keung JW, Bennin KE, Zhang M. Assessing the significant impact of concept drift in software defect prediction. In 43rd annual computer software and applications conference (COMPSAC) 2019(pp. 53-8). IEEE.
[43] Zhang Y, Zeng C, Cheng H, Wang C, Zhang L. Many could be better than all: a novel instance-oriented algorithm for multi-modal multi-label problem. In international conference on multimedia and expo (ICME) 2019 (pp. 838-43). IEEE.
[44] Afric P, Sikic L, Kurdija AS, Silic M. REPD: source code defect prediction as anomaly detection. Journal of Systems and Software. 2020; 168:1-22.
[45] Balogun AO, Basri S, Mahamad S, Abdulkadir SJ, Capretz LF, Imam AA, et al. Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction. Electronics. 2021; 10(2):1-16.
[46] Khurma RA, Alsawalqah H, Aljarah I, Elaziz MA, Damaševičius R. An enhanced evolutionary software defect prediction method using island moth flame optimization. Mathematics. 2021; 9(15):1-20.

Sandeep Ganpat Kamble is doing M. tech. in Computer Science, PCST RGPV Bhopal (MP) and B.E. in Information Technology RGPV Technical University Bhopal (MP). His area of interest are Data mining optimization, Machine learning, Artificial intelligence.
Email:sandeepkamble1506@gmail.com

Animesh Kumar Dubey is working as Assistant professor with the department of Computer Science and Engineering, at Patel College of Science and Technology, Bhopal, India. He has completed his Bachelor of Engineering (B.E.) and M.Tech. degree with Computer Science Engineering from Rajeev Gandhi Technical University, Bhopal (M.P.). He has more than 15 publications in reputed, peer-reviewed national and international journals and conferences. His research areas are Data Mining, Optimization, Machine Learning, Cloud Computing and Artificial Intelligence.
Email:animeshdubey123@gmail.com