e-ISSN : 0976-5166
p-ISSN : 2231-3850

FEATURE AGGLOMERATION BASED FRAMEWORK FOR SOFTWARE BUG PREDICTION

Tamanna

Ph.D. Scholar, Deptt. of CS&E, GJUST,
Hisar, Haryana 125001, India
tamannasharma100 @ gmail.com

Om Prakash Sangwan

Professor, Deptt. of CS&E, GJUST,
Hisar, Haryana 125001, India
sangwan0863 @ gmail.com

Abstract

Growing reliance on software products in the world has put high demand on quality software. Software Bug Prediction is an exercise in enhancing the quality of software by identifying potential bugs or faults in software constructs during the pre-deployment testing phase. Various Machine Learning models have been built to predict faults based on metrics derived from the software. Feature selection and feature reduction (extraction) are two strategies to weed-out redundant and non-useful features thereby reducing the dimensionality of features. The present research study aims to devise an ML model to effectively reduce the dimensionality of the feature-set of data by using a combination of Genetic Algorithm (feature selection) and Feature Agglomeration (feature reduction) without significantly affecting model performance. Datasets containing 62 features are employed to test the model using four classifiers and three cross-validated metrics. Results show that the proposed model reduces the feature-set by more than 80 %. The application of the Kruskal Wallis H test shows the statistical similarity between the results of the proposed model and results without a proposed model. In a majority of test cases, the proposed model has improved the performance of base classifiers by extracting the most informative features.

***Keywords*: Software fault prediction; Dimensionality reduction; Genetic Algorithm; Kruskal Wallis H test.**

1. Introduction

The ever-increasing automation in day-to-day life has led to an exponential rise in highly complex and innovative software products. Efficient testing before deployment of such software systems is mandatory to prevent failures (faults or bugs) and to save precious and already constrained resources. Testing of software for potential faults and their rectification ensures adequate software quality. Comprehensive testing of a newly developed software is resource-intensive and further, it is not to test all possible cases for probable bugs. The goal of software engineers is to optimize the cost and time of software development. Here, Software Fault Prediction (SBP) helps in achieving these goals by predicting code defects during the testing phase itself. SBP can prioritize resources towards the probable faulty software modules and save practitioners from exhaustive testing. SBP can be realized by utilizing Machine Learning (ML) models. Fault prediction models involve data from previous versions for Intra project SBP and historical data from a similar project for inter-project SBP. For prediction purposes, previous studies employed a vast number of classification or supervised methods (labeled data) and unsupervised or clustering (unlabeled data) methods for SBP and showed successful results [Menzies *et al.* (2007)], [Ghotra *et al.* (2015)] ,[Sharma and Sangwan (2022)]. Quality of data has a direct relation to learner's performance. Here data includes features (code metrics, historical defect data and code changes, etc.). The presence of noisy and redundant features degrades the prediction performance by unnecessarily increasing the dimensionality of data.

High dimensional data is cumbersome to process, to visualize and leads to a *curse of dimensionality* [Balogun *et al.* (2019)]. High dimensional data becomes so sparse and variedly dissimilar that extracting meaningful relationships (similarity) among them often becomes computationally difficult. Previous studies in this field utilized two different classes of methods for dealing with the high dimensionality of features: feature selection and feature reduction. Feature Selection (FS) methods (filter and wrapper) help in improving the quality of data by selecting the most prominent features which contribute to prediction performance in a major way [Rathore and Gupta (2014)], [Muthukumaran *et al.* (2015)],[Ghotra *et al.* (2017)] , [Sharma and Sangwan(2021)]. Feature Reduction (FR) techniques, also known as feature extraction, can reduce, combine or make new features [Kondo *et al.* (2019)] from existing features.

In this proposed framework, we utilize powers of both realms (FS and FR) by using the wrapper-based evolutionary optimization method Genetic Algorithm (GA) for feature selection and Feature Agglomeration (FA) for reduction of features. The framework is validated using four classifiers - Naïve Bayes (NB), Random Forest (RF), K-Nearest Neighbor (KNN), and Support Vector Machine (SVM), and five public datasets which contain a fairly large feature-set of sixty-two features as compared to widely used NASA MDP datasets (twenty-one features) with the help of three efficacy measures Accuracy, F1 score, and AUC-ROC (Area under Receiver Operating Characteristics).

The present study is organized as follows: Section 2 is a literature survey of studies that utilize GA and FA in SBP models. Section 3 details the dataset, classification techniques, experimental setup, and methodology employed for this experiment. Section 4 consists of results and discussion and Section 5 provides conclusions and future directions.

2. Related Work

Past studies in this field have invariably employed GA as a feature selector using different parameter settings [Alsghaier *et al.* (2020)] proposed approach which integrates GA and Particle Swarm Optimization (PSO) with reciprocal integration and employed with Support Vector Machine on twenty-four datasets (12 *java* and 12 NASA datasets). Results based on efficacy measures like precision, recall, error rate, accuracy, F-score, specificity, and standard deviation showed improved performance on large- and small-scale datasets. [Ayon and Islam (2019)] also employed GA with PSO but differently: first, GA is used for feature selection, and second, PSO is applied for making clusters of selected features. These clusters are used for training four different Neural Networks (NN): feedforward NN, Recurrent NN, Artificial NN, and Deep NN on four datasets of the NASA software repository. The accuracy score-based evaluation metric showed the best results on deep neural networks in comparison with four other research studies. [Sarro *et al.* (2012)] utilized GA with SVM for fault prediction on five inter-release software systems with the help of Recall, Precision, and F-measure. This study concludes that GA in combination with SVM behaves consistently and outperforms LR, MLP, C4.5, KNN, and RF classifiers in terms of F-measure and Recall. [Martino *et al.*, (2011)] also utilized GA for configuring SVM on jEdit data from the PROMISE repository and analyzed performance for both inter-release and intra-release projects. Compared with baseline models SVM-Grid, LR, NB, C4.5, MLP, RF, and KNN, this model (GA SVM combined and F-measure as fitness function) improves recall score without compromising precision score. [Fazel and Sadat (2016)] utilized GA for predicting software errors and results in best conditions showed recognition rates of more than 95 percent on 13 datasets of the NASA software fault repository. [Turabieh *et al.* (2019)] applied binary GA, binary PSO, and binary ant colony optimization as feature-selectors for enhancing the effectiveness of a layered RNN (Recurrent Neural Network). Performance is assessed on nineteen software projects taken from the Promise software repository and compared with five state-of-the-art classifiers in terms of AUC-ROC as an evaluation metric. Results concluded that a layered RNN with iterated feature selector outperforms NB, DT, ANN, LR, and KNN in terms of AUC-ROC score and also claimed that a selection of features plays an important role in the building of a high-quality SBP model. [Kumar *et al.* (2016)] employed GA in association with logistic regression, extreme learning, and three variations of support vector machine based on kernel functions (polynomial, linear and radial basis), and results are validated on thirty Java open-source projects. The t-test-based statistical test reveals that there is no change in performance even after a reduced set of features and the proposed cost analysis model suggests that the fault prediction model is suitable for Software datasets having data imbalance between 36. 3% to 46.4%. [Kondo *et al.* (2019)] studied the impact of eight feature reduction techniques (Principal Component Analysis, Fast Map, Feature Agglomeration, Transfer Component Analysis, Random Projection, Restricted Boltzmann Machine, and Autoencoder) and two feature selection techniques (correlation-based FS and Consistency based FS) on five supervised (LR, DT, RF, NB, and logistic model tree) and five unsupervised (K-means, fuzzy C-means, spectral clustering, partition around medoids, and neural gas) defect prediction models. Results have been analyzed based on AUC and conclude that Neural Network-based feature reduction methods give the best performance for unsupervised models, and feature selection/ generation enhances the efficiency of almost all the studied models in comparison with original features.

Our framework is unique since it is a combination of GA as FS and FA as FR. No previous study to our best knowledge has adopted such an approach.

3. Methodology

The overall method (Fig. 1) for this experimental study was implemented in *Python* language (ver. 3.8) using *Spyder* platform (ver. 4.0) and *sklearn* (ver. 0.24) library [Pedregosa *et al.* (2011)].

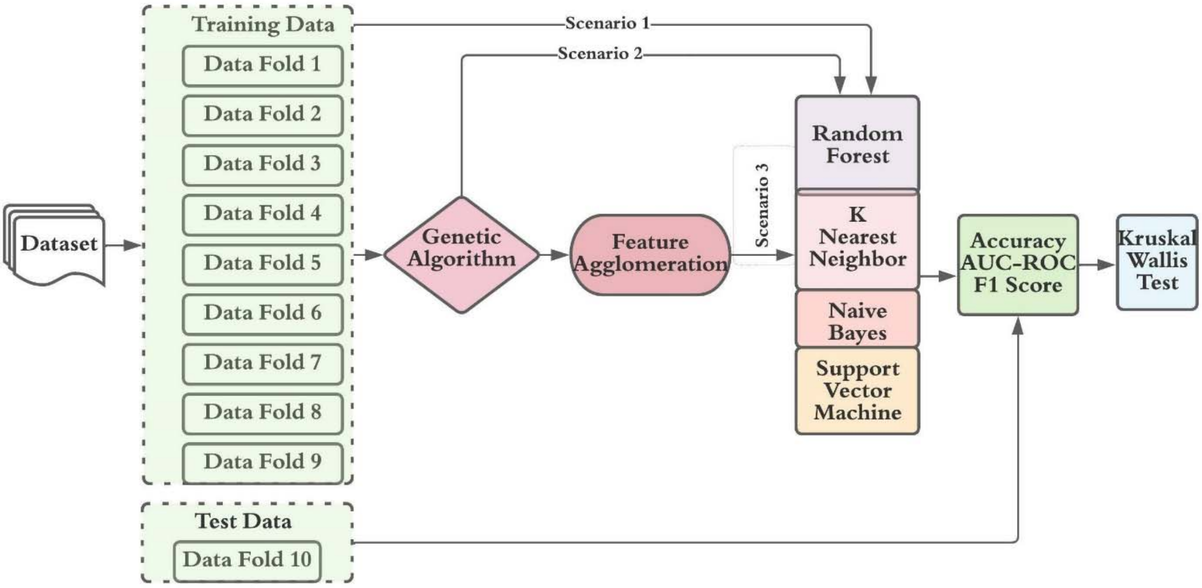

Fig. 1. Methodology

3.1. *Dataset*

Six datasets (Table 1) containing class-level data from the AEEEM software repository provided by [D'Ambros *et al.* (2010)] are used in this study. Each of the datasets has 62 attributes including the label attribute. Fault percentage varies from 9.2% (Lucene) to 39.8% (EQ) i.e., datasets are highly imbalanced.

Table 1. AEEEM Datasets

Dataset	No. of instances	No. of attributes	Fault percentage(%)
EQ	324	62	39.8
PDE	1497	62	13.9
JDT	997	62	20.6
LUCENE	691	62	9.2
ML	1862	62	13.1

AEEEM datasets are public and contain static code metrics such as O-O (Object-Oriented) metrics [Basili *et al.* (1996)], C-K (Chidamber-Kemerer) metrics [Chidamber and Kemerer (1994)], the churn of C-K and O-O metrics, the entropy of C-K and O-O metrics, and change features.

3.2. *Preprocessing*

The quality of data directly impacts the efficiency of an ML model. It is assumed that data employed in ML models are identically distributed and independent of each other. But the data extracted from various sources are seldom as assumed. Hence comes the role of data preprocessing.
Data preprocessing involves steps to clean datasets such as removing duplicate instances, treating absent values, standardizing, normalizing, imbalance treatment, etc.
Data standardization in this study has been done using MinMaxScaler in the (0,1) range based on Eq. (1). This transformation is an alternative to Zero mean and unit variance scaling.

$$X_{scaled} = \frac{X - X_{min}(axis=0)}{X_{max}(axis=0) - X_{min}(axis=0)} \qquad (1)$$

To tackle data imbalance, minority samples are over-sampled. Undersampling leads to the loss of data the criticality of which is unknown. SMOTE i.e. Synthetic Minority Over-sampling Technique [Chawla *et al.* (2002)] is employed for oversampling. The additional minority samples are randomly generated after combining the locations of any number of samples in the neighborhood (user-defined) vicinity.

3.3. *Attribute Selection and reduction*

High dimensionality affects the performance of ML models adversely. It jeopardizes the curve-fitting process of the ML model. Since our datasets have 62 features, we have devised a combination of Genetic Algorithm(GA) i.e. feature selection, and Feature Agglomeration (FA) i.e. feature reduction strategy to reduce

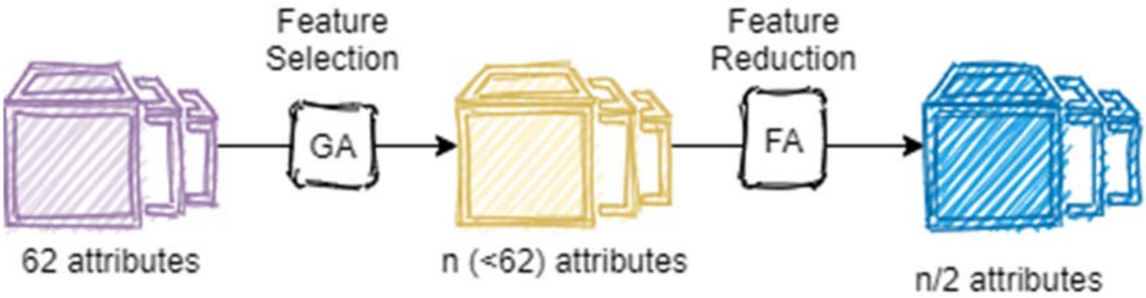

Fig. 2. Attribute Selection and Reduction

the attributes.
GA [Holland *et al.* (1978)] is an evolutionary stochastic technique based on Darwin's Evolution Theory. It follows

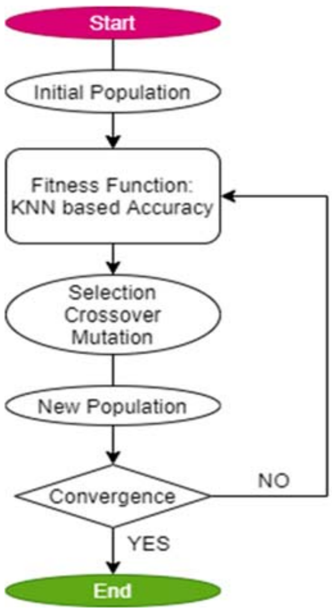

Fig. 3. GA

process of natural selection where the fittest individuals (based on chosen Fitness Function) are selected for generating offspring (using Crossover and Mutation) for the next generation. A simplified flowchart of GA optimization is shown in Fig.3. We have used GA based wrapper which includes K-nearest Neighbour (KNN) Classifier accuracy-based fitness function.
The crossover rate (probability) determines the chance that two chromosomes exchange their constituents. Mutation rate (probability) determines the number of chromosomes that should change in one generation. Mutation rate prevents algorithm to converge at local optima. The initial population sets the search space for the algorithm.
The parameters of GA are chosen such that the algorithm converges faster [Grefenstetter and John (1986)]:

- Mutation rate - 0.01
- Initial population - 50
- No. of Iterations - 150
- Cross-over rate - 0.95
- No. of runs - 30

A *fitness vs. iterations* graph from our study showing Run No. 6 (out of a total of 30 runs) of the Lucene dataset is shown in Fig.4.
Results of the GA wrapper are taken as the average of results of 30 runs.

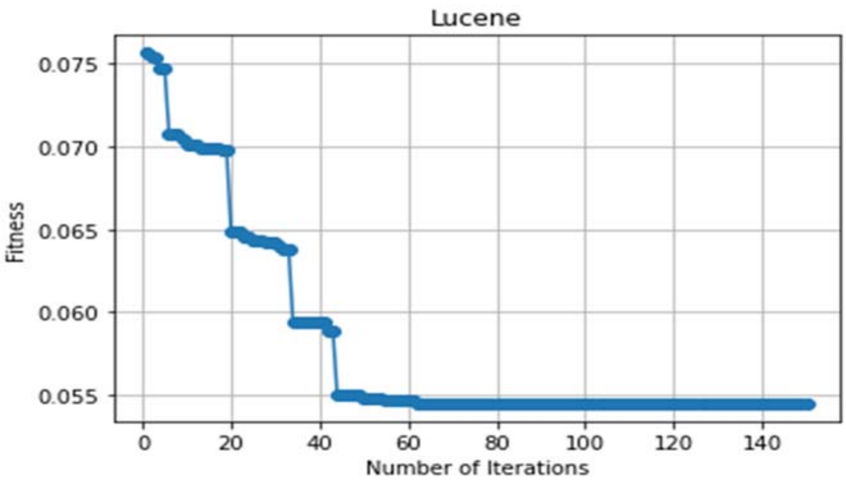

Fig. 4. Fitness vs Iteration (*Run No. 6*, '*Lucene'*)

FA is a hierarchical clustering strategy that pools together similar features. Features are reduced based on a specified criterion, here we have used the maximum distance between observations of pairs of clusters.

3.4. *Scenarios*

Three scenarios (Fig. 1) are considered to compare the proposed attribute selection and reduction technique:

- *Scenario 1*: original attributes
- *Scenario 2*: attributes selected by GA
- *Scenario 3*: attributes after feature selection by GA and subsequent feature reduction by FA. FA reduced features to half the number of features selected by GA.

3.5. *Classifiers*

Four ML classifiers from different classes- Random Forest (RF) [Breimen (2001)] (Ensemble class), K-Nearest Neighbor (KNN) [Altman (1992)] (Instance-based), Naïve Bayes (NB) [Zhang (2005)] (probabilistic class) and Support Vector Machine (SVM) are fitted with obtained attributes from above-referred Scenarios.

3.6. *Performance Metrics*

Accuracy, AUC-ROC, and F1 Score are evaluated to compare classifier performance. Accuracy is defined as ratio of correctly classified target labels to total classified labels.
The Area under Curve (AUC) is a single number that summarizes the Receiver Operating Characteristics (ROC) of a classifier. ROC is a graph between the TPR (True Positive Rate) and FPR (False Positive Rate) at different decision boundary thresholds.F1 Score is the weighted average of Precision and Recall Eq. (2)

$$F1\ Score = 2\ \frac{Precision * Recall}{Precision + Recall} \tag{2}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4}$$

True Positives are target faulty labels correctly classified as such whereas False Positives are non-faulty target labels classified as faulty. False Negatives are faulty target labels classified as non-faulty. These parameters are calculated using a Confusion Matrix (Fig. 5).

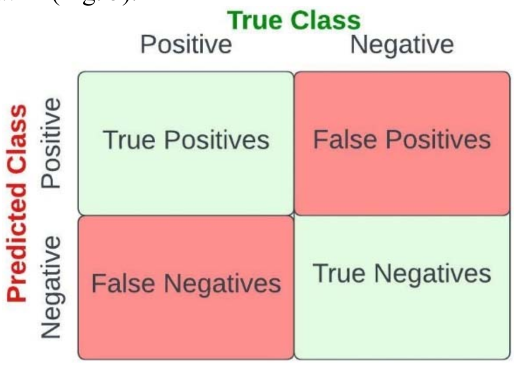

Fig. 5. Confusion Matrix

Higher Precision and recall values indicate better performance of the ML classifier.

3.7. *Cross-Validation*

Cross-validation (CV) of an ML model leads to generalized results with lower bias as compared to any random run of the model. We have employed a Stratified 10-fold CV that creates 10 train and test folds (proportion of target classes preserved in each fold) upon which ML models are trained and performance metrics calculated. Average is taken for the 10 metric values so obtained respectively for comparison.

4. Results and Discussion

4.1. *Scenario 1*

Under scenario 1, a dataset with original attributes, after preprocessing, was fed to ML Classifiers and Stratified with 10-fold cross-validated Accuracy, AUC-ROC, and F1 Score calculated. present The Accuracy, ROC-AUC, and F1 Score obtained in this scenario are shown in Table 2, Table 3, and Table 4 respectively.

Table 2 Accuracy

Classifier	EQ	JDT	Lucene	ML	PDE
NB	0.7253	0.8395	0.8496	0.8276	0.8375
KNN	0.6820	0.7864	0.7670	0.7492	0.7804
RF	**0.7933**	**0.8476**	**0.8871**	**0.8716**	**0.8526**
SVC	0.7497	0.8225	0.8090	0.7207	0.8145

Table 3 AUC-ROC

Classifier	EQ	JDT	Lucene	ML	PDE
NB	0.8129	0.8181	**0.7860**	0.7305	0.8159
KNN	0.7554	0.8365	0.7211	0.7579	0.8222
RF	**0.8531**	**0.8817**	0.7828	**0.8248**	**0.8790**
SVC	0.8076	0.8600	0.7482	0.7792	0.8593

Table 4. F1 Score

Classifier	EQ	JDT	Lucene	ML	PDE
NB	0.6881	0.7362	**0.6630**	0.6343	0.7374
KNN	0.6784	0.7014	0.5760	0.6276	0.7141
RF	**0.7839**	**0.7686**	0.6596	**0.7100**	**0.7704**
SVC	0.7457	0.7531	0.5681	0.6045	0.7502

Random Forest (RF) has outperformed the other three classifiers in all the evaluated metrics except NB AUC-ROC and F1 Score in Lucene Dataset.

4.2. *Scenario 2*

In Scenario 2, attributes selected from the GA wrapper are taken and performance metrics evaluated. Table 5, Table 6, and Table 7 present the Accuracy, ROC-AUC, and F1 Scores obtained in this scenario respectively.

Table 5. Accuracy

Classifier	EQ	JDT	Lucene	ML	PDE
NB	0.7222	0.8335	0.8481	0.8319	0.8385
KNN	0.7689	0.7984	0.7915	0.7481	0.7894
RF	0.7933	0.8425	0.8929	0.8663	0.8445
SVC	0.7341	0.8325	0.8060	0.7400	0.8134

Table 6. AUC-ROC

Classifier	EQ	JDT	Lucene	ML	PDE
NB	0.8157	0.8259	**0.8029**	0.7457	0.8307
KNN	0.8263	0.8278	0.7370	0.7345	0.8215
RF	**0.8465**	0.8577	0.7866	**0.8046**	0.8619
SVC	0.8261	**0.8585**	0.7847	0.7506	**0.8630**

Table 7. F1 Score

Classifier	EQ	JDT	Lucene	ML	PDE
NB	0.6838	0.7339	**0.6647**	0.6284	0.7344
KNN	0.7436	0.7323	0.6078	0.6189	0.7179
RF	**0.7901**	0.7616	0.6587	**0.6983**	0.7407
SVC	0.7491	**0.7650**	0.6030	0.6012	**0.7483**

RF classifier has outperformed other classifiers in terms of AUC-ROC, Accuracy, and F1 Score ((EQ and ML datasets). NB has performed best in Lucene Dataset based on F1 Score and AUC-ROC. SVC has performed best in PDE and JDT datasets based on F1 Score and AUC-ROC.

4.3. *Scenario 3*

In Scenario 3, optimum attributes selected from the GA wrapper are further reduced to half by application of FA. Fig. 6, and 7,8,9 are boxplots of all metrics in all three scenarios classifier-wise. Boxplot legends with "*_orig*" extension are Scenario 1 results, with *"_GA"* are Scenario 2 results and with *"_GF"* are Scenario 3 results.

It can be observed from the boxplots that performance in Scenario 3 has increased for NB and KNN classifiers. RF and SVM classifiers have given equivalent performance in Scenario 2 and Scenario 3.

Further, it can be seen that boxplots for Scenario 3 have higher median values. It can be said that Scenario 3 in general has

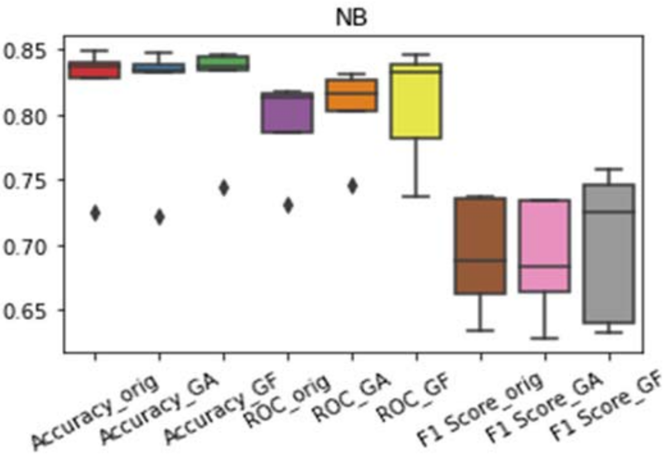

Fig. 6. NB Boxplot

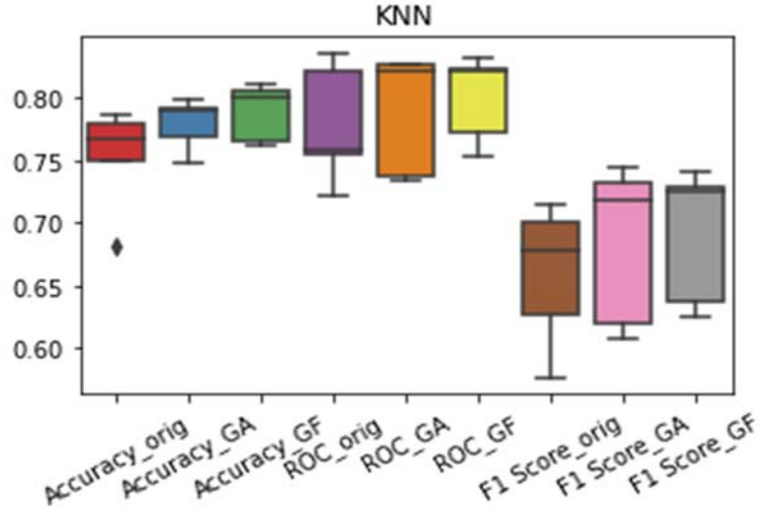

Fig. 7. KNN Boxplot

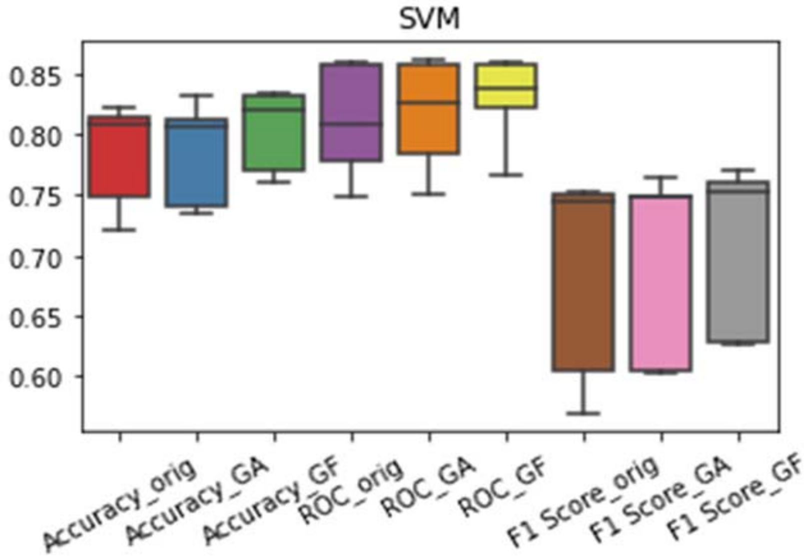

Fig. 8. SVC Boxplot

improved performance as compared to Scenario 1 and Scenario 2.

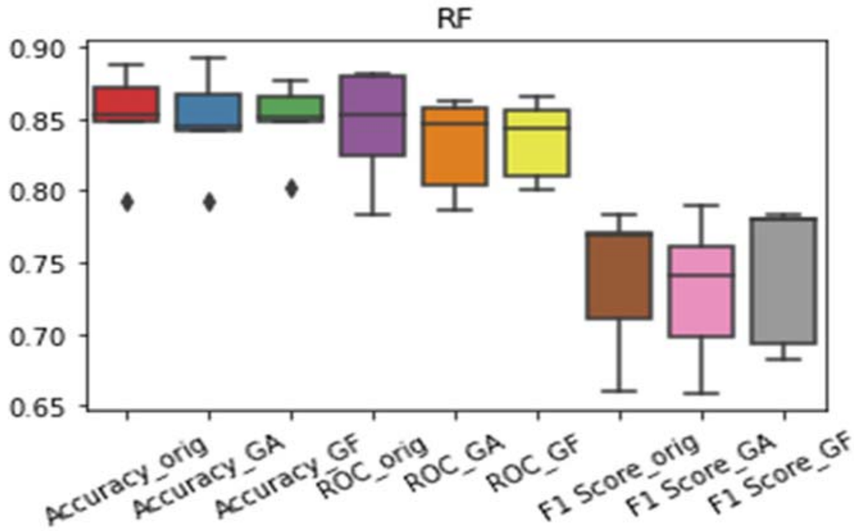

Fig. 9. RF Boxplot

Attributes in the three scenarios and overall dimensionality reduction are given in Table 8.

Table 8. Dimensionality Reduction

Dataset	Scenario 1 Attributes	Scenario 2 Attributes (n)	Scenario 3 Attributes (n/2)	Dimensionality reduction (%)
EQ	61	20	10	83.60
JDT	61	21	10	83.60
Lucene	61	11	5	91.80
ML	61	25	12	80.32
PDE	61	22	11	81.96

4.4. *Statistical significance*

For statistical validation of the results obtained above, we assume null and alternate hypotheses as:

$H_{o:}$ There is no difference in performance

$H_{a:}$ There is a difference in performance

We applied *Kruskal Wallis H Test* to know if the results between three scenarios are statistically different or not. We compared all three-performance metrics in the three scenarios. P-values are summarized in Table 9.

Table 9. p-values

Dataset	Accuracy	AUC-ROC	F1 Score
	Scenario 1, Scenario 2, Scenario 3		
EQ	0.45	0.245	0.793
JDT	0.912	0.874	0.925
Lucene	0.98	0.389	0.735
ML	0.793	0.693	0.793
PDE	0.778	0.778	0.33

Considering threshold α = 0.05, the Null Hypothesis is accepted and statistical difference between performances in Scenario 1, Scenario 2, and Scenario 3 is insignificant.
Analyzing Table VIII along with the boxplots and Table IX, it can be inferred that our framework has been able to maintain the performance of SBP even after reducing the dimensionality by more than 80%.

5. Threats to Validity

Internal Validity: Baseline classifiers used in this study are not hyperparameter tuned. Hyperparameter tuning will increase the absolute value of results but the trends for different datasets and different metrics will generally remain the same.
External Validity: The study is conducted on AEEEM datasets because they are public and contain varied 62 attributes. Application of this framework to other datasets might yield different results leading to enhanced generalization.

6. Conclusions

We devised a Feature Agglomeration based framework for SBP using GA as an intermediate feature selector. We found that our framework on AEEEM datasets is practical since it leads to a huge reduction (more than 80%) in attributes without compromising performance based on Accuracy, AUC-ROC, and F1 Score metrics. Further

- This proves the assertion that all attributes are not important for an efficient ML model. Appropriate feature selection and feature reduction techniques should be used to select or extract significant features.
- The application of this framework has improved the performance of baseline classifiers (RF, KNN, SVM, NB) by creating meaningful features amongst which relationships can be deduced.

This framework, GA followed by FA, can be gainfully employed in situations where dimensionality reduction without performance is a key requirement.

Conflicts of interest

"The authors have no conflicts of interest to declare"

REFERENCES

[1] Balogun, Abdullateef Oluwagbemiga, Said Jadid Abdulkadir, Shuib Basri, and Ahmad Sobri Hashim. "Performance analysis of feature selection methods in software defect prediction: a search method approach." Applied Sciences 9, no. 13 (2019): 2764.
[2] Alsghaier, Hiba, and Mohammed Akour. "Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier." *Software: Practice and Experience* 50, no. 4 (2020): 407-427.
[3] Altman, Naomi S. "An introduction to kernel and nearest-neighbor nonparametric regression." *The American Statistician 46*, no. 3 (1992): 175-185.
[4] Ayon, Safial Islam. "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, 2019.
[5] Balogun, Abdullateef O., Shuib Basri, Said A. Jadid, Saipunidzam Mahamad, Malek A. Al-momani, Amos O. Bajeh, and Ammar K. Alazzawi. "Search-based wrapper feature selection methods in software defect prediction: an empirical analysis." In *Computer Science On-line Conference*, pp. 492-503. Springer, Cham, 2020.
[6] Balogun, Abdullateef O., Shuib Basri, Saipunidzam Mahamad, Said J. Abdulkadir, Malek A. Almomani, Victor E. Adeyemo, Qasem Al-Tashi, Hammed A. Mojeed, Abdullahi A. Imam, and Amos O. Bajeh. "Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study." *Symmetry* 12, no. 7 (2020): 1147.
[7] Basili, V.R., Briand, L.C., Melo, W.L.: A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering (TSE) 22(10), 751–761 (1996)
[8] Breiman, Leo. "Random forests." *Machine learning* 45, no. 1 (2001): 5-32.
[9] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." *Journal of artificial intelligence research 16* (2002): 321-357.
[10] Chidamber, S.R., Kemerer, C.F.: A metrics suite for object-oriented design. IEEE Transactions on Software Engineering (TSE) 20(6), 476–493 (1994)
[11] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "An extensive comparison of bug prediction approaches." In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 31-41. IEEE, 2010.
[12] Di Martino, Sergio, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. "A genetic algorithm to configure support vector machines for predicting fault-prone components." In *International conference on product focused software process improvement*, pp. 247-261. Springer, Berlin, Heidelberg, 2011.
[13] Fazel, Fahimeh Sadat. "A new method to predict the software fault using improved genetic algorithm." Bulletin de la Société Royale des Sciences de Liège 85 (2016): 187-202.
[14] Gao, Kehan, Taghi M. Khoshgoftaar, and Naeem Seliya. "Predicting high-risk program modules by selecting the right software measurements." *Software Quality Journal* 20, no. 1 (2012): 3-42.
[15] Gao, Kehan, Taghi M. Khoshgoftaar, Huanjing Wang, and Naeem Seliya. "Choosing software metrics for defect prediction: an investigation on feature selection techniques." *Software: Practice and Experience* 41, no. 5 (2011): 579-606.
[16] Ghotra, Baljinder, Shane McIntosh, and Ahmed E. Hassan. "A large-scale study of the impact of feature selection techniques on defect classification models." In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), pp. 146-157. IEEE, 2017.
[17] Ghotra, Baljinder, Shane McIntosh, and Ahmed E. Hassan. "Revisiting the impact of classification techniques on the performance of defect prediction models." In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 1, pp. 789-800. IEEE, 2015.
[18] Grefenstette, John J. "Optimization of control parameters for genetic algorithms." *IEEE Transactions on systems, man, and cybernetics* 16, no. 1 (1986): 122-128.

[19] Guyon, Isabelle, and André Elisseeff. "An introduction to variable and feature selection." *Journal of machine learning research* 3, no. Mar (2003): 1157-1182.
[20] Hassouneh, Yousef, Hamza Turabieh, Thaer Thaher, Iyad Tumar, Hamouda Chantar, and Jingwei Too. "Boosted whale optimization algorithm with natural selection operators for software fault prediction." *IEEE Access 9* (2021): 14239-14258.
[21] Holland, John H., and Judith S. Reitman. "Cognitive systems based on adaptive algorithms." In *Pattern-directed inference systems*, pp. 313-329. Academic Press, 1978.
[22] Huang, Jin, and Charles X. Ling. "Using AUC and accuracy in evaluating learning algorithms." *IEEE Transactions on knowledge and Data Engineering* 17, no. 3 (2005): 299-310.
[23] Kondo, Masanari, Cor-Paul Bezemer, Yasutaka Kamei, Ahmed E. Hassan, and Osamu Mizuno. "The impact of feature reduction techniques on defect prediction models." *Empirical Software Engineering* 24, no. 4 (2019): 1925-1963.
[24] Kumar, Lov, and Santanu Ku Rath. "Application of genetic algorithm as feature selection technique in development of effective fault prediction model." In 2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON), pp. 432-437. IEEE, 2016.
[25] Mabayoje, M.A., Balogun, A.O., Bello, S.M., Atoyebi, J.O., Mojeed, H.A., Ekundayo, A.H.:Wrapper feature selection based heterogeneous classifiers for software defect prediction.Adeleke *Univ. J. Eng. Technol.2*, 1–11 (2019)
[26] Menzies, Tim, Jeremy Greenwald, and Art Frank. "Data mining static code attributes to learn defect predictors." IEEE transactions on software engineering 33, no. 1 (2006): 2-13.
[27] Muthukumaran, K., Akhila Rallapalli, and NL Bhanu Murthy. "Impact of feature selection techniques on bug prediction models." In Proceedings of the 8th India Software Engineering Conference, pp. 120-129. 2015.
[28] Muthukumaran, K., Akhila Rallapalli, and NL Bhanu Murthy. "Impact of feature selection techniques on bug prediction models." In *Proceedings of the 8th India Software Engineering Conference*, pp. 120-129. 2015.
[29] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel *et al.* "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* 12 (2011): 2825-2830.
[30] R. S. Wahono, N. Suryana, and S. Ahmad, ''Metaheuristic optimization based feature selection for software defect prediction,'' *J. Softw.*, vol. 9, no. 5, pp. 1324–1333, May 2014.
[31] Raheem, Muiz, Ahmed Ameen, Falilat Ayinla, and Bolaji Ayeyemi. "Software Defect Prediction Using Metaheuristic Algorithms and Classification Techniques." *Ilorin Journal of Computer Science and Information Technology 3*, no. 1 (2020): 23-39.
[32] Rathore, Santosh Singh, and Atul Gupta. "A comparative study of feature-ranking and feature-subset selection techniques for improved fault prediction." In Proceedings of the 7th India Software Engineering Conference, pp. 1-10. 2014.
[33] Sarro, Federica, Sergio Di Martino, Filomena Ferrucci, and Carmine Gravino. "A further analysis on the use of genetic algorithm to configure support vector machines for inter-release fault prediction." In *Proceedings of the 27th annual ACM symposium on applied computing*, pp. 1215-1220. 2012.
[34] Sharma, Tamanna, and Om Prakash Sangwan. "Sine-Cosine Algorithm for Software Fault Prediction." In 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 701-706. IEEE, 2021.
[35] Sharma, Tamanna, and Om Prakash Sangwan. "Random Permutation-based Hybrid Feature Selection for Software Bug Prediction using Bayesian Statistical Validation." International Journal of Engineering Trends and Technology 70 (2022): 188-202.
[36] Tantithamthavorn, Chakkrit, Shane McIntosh, Ahmed E. Hassan, and Kenichi Matsumoto. "Automated parameter optimization of classification techniques for defect prediction models." In *Proceedings of the 38th International Conference on Software Engineering*, pp. 321-332. 2016.
[37] Thaher, Thaer, Majdi Mafarja, Baker Abdalhaq, and Hamouda Chantar. "Wrapper-based feature selection for imbalanced data using binary queuing search algorithm." In *2019 2nd international conference on new trends in computing sciences (ICTCS)*, pp. 1-6. IEEE, 2019.
[38] Turabieh, Hamza, Majdi Mafarja, and Xiaodong Li. "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction." Expert systems with applications 122 (2019): 27-42.
[39] Xu, Zhou, Jin Liu, Zijiang Yang, Gege An, and Xiangyang Jia. "The impact of feature selection on defect prediction performance: An empirical comparison." In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 309-320. IEEE, 2016.
[40] Yu, Lei, and Huan Liu. "Feature selection for high-dimensional data: A fast correlation-based filter solution." In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 856-863. 2003.
[41] Zhang, Harry. "Exploring conditions for the optimality of naive Bayes." *International Journal of Pattern Recognition and Artificial Intelligence 19*, no. 02 (2005): 183-198.

Authors Profile

Tamanna is currently pursuing a Ph.D. in Software Engineering and Soft Computing from Guru Jambheshwar University of Science and Technology, Hisar, Haryana. She received an M.Tech. in Computer Science and Engineering from Banasthali University, Rajasthan. Her area of research includes Software Engineering with Machine Learning, Mining Software Repositories, Software Reliability engineering, and Automated Software Debugging.

Om Prakash Sangwan is currently working as Professor, Deptt. of Computer Science & Engineering, Guru Jamheshwar University of Science and Technology, Hisar. He received Ph.D. and M.Tech. in Computer Science & Engineering from Guru Jambheshwar University of Science &Technology, Hisar, Haryana. His research area includes Software Engineering focusing on Planning, Designing, Testing, Software Metrics, Neural Networks, Neuro-Fuzzy, and Fuzzy Logic.