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)].

![Methodology Diagram](image)

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>
<thead>
<tr>
<th>Dataset</th>
<th>No. of instances</th>
<th>No. of attributes</th>
<th>Fault percentage(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>324</td>
<td>62</td>
<td>39.8</td>
</tr>
<tr>
<td>PDE</td>
<td>1497</td>
<td>62</td>
<td>13.9</td>
</tr>
<tr>
<td>JDT</td>
<td>997</td>
<td>62</td>
<td>20.6</td>
</tr>
<tr>
<td>LUCENE</td>
<td>691</td>
<td>62</td>
<td>9.2</td>
</tr>
<tr>
<td>ML</td>
<td>1862</td>
<td>62</td>
<td>13.1</td>
</tr>
</tbody>
</table>

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.

\[
x_{\text{scaled}} = \frac{x - x_{\text{min}}(\text{axis}=0)}{x_{\text{max}}(\text{axis}=0) - x_{\text{min}}(\text{axis}=0)} \tag{1}
\]
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

![Attribute Selection and Reduction](image1)

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

![GA](image2)

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), Naive 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)

\[
F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2)
\]

\[
Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3)
\]

\[
Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (4)
\]
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).

![Confusion Matrix](image)

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. The Accuracy, ROC-AUC, and F1 Score obtained in this scenario are shown in Table 2, Table 3, and Table 4 respectively.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>EQ</th>
<th>JDT</th>
<th>Lucene</th>
<th>ML</th>
<th>PDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.7253</td>
<td>0.8395</td>
<td>0.8496</td>
<td>0.8276</td>
<td>0.8375</td>
</tr>
<tr>
<td>KNN</td>
<td>0.6820</td>
<td>0.7864</td>
<td>0.7670</td>
<td>0.7492</td>
<td>0.7804</td>
</tr>
<tr>
<td>RF</td>
<td>0.7933</td>
<td>0.8476</td>
<td>0.8871</td>
<td>0.8716</td>
<td>0.8526</td>
</tr>
<tr>
<td>SVC</td>
<td>0.7497</td>
<td>0.8225</td>
<td>0.8090</td>
<td>0.7207</td>
<td>0.8145</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier</th>
<th>EQ</th>
<th>JDT</th>
<th>Lucene</th>
<th>ML</th>
<th>PDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.8129</td>
<td>0.8181</td>
<td>0.7860</td>
<td>0.7305</td>
<td>0.8159</td>
</tr>
<tr>
<td>KNN</td>
<td>0.7554</td>
<td>0.8365</td>
<td>0.7211</td>
<td>0.7579</td>
<td>0.8222</td>
</tr>
<tr>
<td>RF</td>
<td>0.8531</td>
<td>0.8817</td>
<td>0.7828</td>
<td>0.8248</td>
<td>0.8790</td>
</tr>
<tr>
<td>SVC</td>
<td>0.8076</td>
<td>0.8600</td>
<td>0.7842</td>
<td>0.7792</td>
<td>0.8593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier</th>
<th>EQ</th>
<th>JDT</th>
<th>Lucene</th>
<th>ML</th>
<th>PDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.6881</td>
<td>0.7362</td>
<td>0.6630</td>
<td>0.6343</td>
<td>0.7374</td>
</tr>
<tr>
<td>KNN</td>
<td>0.6784</td>
<td>0.7014</td>
<td>0.5760</td>
<td>0.6276</td>
<td>0.7141</td>
</tr>
<tr>
<td>RF</td>
<td>0.7839</td>
<td>0.7686</td>
<td>0.6596</td>
<td>0.7100</td>
<td>0.7704</td>
</tr>
<tr>
<td>SVC</td>
<td>0.7457</td>
<td>0.7531</td>
<td>0.5681</td>
<td>0.6045</td>
<td>0.7502</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Classifier</th>
<th>EQ</th>
<th>JDT</th>
<th>Lucene</th>
<th>ML</th>
<th>PDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.7222</td>
<td>0.8335</td>
<td>0.8481</td>
<td>0.8319</td>
<td>0.8385</td>
</tr>
<tr>
<td>KNN</td>
<td>0.7689</td>
<td>0.7984</td>
<td>0.7915</td>
<td>0.7481</td>
<td>0.7894</td>
</tr>
<tr>
<td>RF</td>
<td>0.7933</td>
<td>0.8425</td>
<td>0.8929</td>
<td>0.8663</td>
<td>0.8445</td>
</tr>
<tr>
<td>SVC</td>
<td>0.7341</td>
<td>0.8325</td>
<td>0.8060</td>
<td>0.7400</td>
<td>0.8134</td>
</tr>
</tbody>
</table>

Table 6. AUC-ROC

<table>
<thead>
<tr>
<th>Classifier</th>
<th>EQ</th>
<th>JDT</th>
<th>Lucene</th>
<th>ML</th>
<th>PDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.8157</td>
<td>0.8259</td>
<td><strong>0.8029</strong></td>
<td>0.7457</td>
<td>0.8307</td>
</tr>
<tr>
<td>KNN</td>
<td>0.8263</td>
<td>0.8278</td>
<td>0.7370</td>
<td>0.7345</td>
<td>0.8215</td>
</tr>
<tr>
<td>RF</td>
<td><strong>0.8465</strong></td>
<td>0.8577</td>
<td>0.7866</td>
<td><strong>0.8046</strong></td>
<td>0.8619</td>
</tr>
<tr>
<td>SVC</td>
<td>0.8261</td>
<td><strong>0.8585</strong></td>
<td>0.7847</td>
<td>0.7506</td>
<td><strong>0.8630</strong></td>
</tr>
</tbody>
</table>

Table 7. F1 Score

<table>
<thead>
<tr>
<th>Classifier</th>
<th>EQ</th>
<th>JDT</th>
<th>Lucene</th>
<th>ML</th>
<th>PDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.6838</td>
<td>0.7339</td>
<td><strong>0.6647</strong></td>
<td>0.6284</td>
<td>0.7344</td>
</tr>
<tr>
<td>KNN</td>
<td>0.7436</td>
<td>0.7323</td>
<td>0.6078</td>
<td>0.6189</td>
<td>0.7179</td>
</tr>
<tr>
<td>RF</td>
<td><strong>0.7901</strong></td>
<td>0.7616</td>
<td>0.6587</td>
<td><strong>0.6983</strong></td>
<td>0.7407</td>
</tr>
<tr>
<td>SVC</td>
<td>0.7491</td>
<td><strong>0.7650</strong></td>
<td>0.6630</td>
<td>0.6012</td>
<td><strong>0.7483</strong></td>
</tr>
</tbody>
</table>

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.

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

**Table 8. Dimensionality Reduction**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Scenario 1 Attributes (n)</th>
<th>Scenario 2 Attributes (n/2)</th>
<th>Scenario 3 Attributes (n/2)</th>
<th>Dimensionality reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>61</td>
<td>20</td>
<td>10</td>
<td>83.60</td>
</tr>
<tr>
<td>JDT</td>
<td>61</td>
<td>21</td>
<td>10</td>
<td>83.60</td>
</tr>
<tr>
<td>Lucene</td>
<td>61</td>
<td>11</td>
<td>5</td>
<td>91.80</td>
</tr>
<tr>
<td>ML</td>
<td>61</td>
<td>25</td>
<td>12</td>
<td>80.32</td>
</tr>
<tr>
<td>PDE</td>
<td>61</td>
<td>22</td>
<td>11</td>
<td>81.96</td>
</tr>
</tbody>
</table>

**4.4. Statistical significance**

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

- $H_0$: 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**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>AUC-ROC</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>0.45</td>
<td>0.245</td>
<td>0.793</td>
</tr>
<tr>
<td>JDT</td>
<td>0.912</td>
<td>0.874</td>
<td>0.925</td>
</tr>
<tr>
<td>Lucene</td>
<td>0.98</td>
<td>0.389</td>
<td>0.735</td>
</tr>
<tr>
<td>ML</td>
<td>0.793</td>
<td>0.693</td>
<td>0.793</td>
</tr>
<tr>
<td>PDE</td>
<td>0.778</td>
<td>0.777</td>
<td>0.33</td>
</tr>
</tbody>
</table>
Considering threshold $\alpha = 0.05$, 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


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 Jambheshwar 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.