Received: July 18, 2020. Revised: November 5, 2020.

INASS

*International Journal of*
Intelligent Engineering & Systems

http://www.inass.org/

# A Bacterial Foraging Algorithm with Random Forest Classifier for Detecting the Design Patterns in Source Code

**Srinivasa Suresh Sikhakolli[1]*** **Asha Kiran Sikhakolli[2]**

[1]*Faculty of Engineering Science, Vishwakarma University, Laxmi Nagar, Kondhwa, India*
[2]*Department of MBA, Rajarshi Shahu College of Engineering, Tathawade, India*
* Corresponding author's Email: sssuresh74@gmail.com

**Abstract:** A program design can be understood by the developers through detecting the design patterns in object-oriented programming source code. The main advantage of the design pattern detection includes maintainability, understand-ability and reusability of object-oriented programs. An object-oriented program is used as the input to the design pattern detection techniques, which judges the candidate roles of the patterns. The existing pattern detection techniques use the machine learning (ML) techniques, namely Support Vector Machine (SVM) and Decision Tree with all feature metrics (more than 70 metrics) to identify patterns, which increases the computation time. To minimize the issues of using all feature metrics, an effective feature selection technique is used in the research study. The most important relevant features are selected by proposed Bacterial Foraging Algorithm (BFA) and given as input to ensemble classifiers namely SVM, Decision Tree and Random Forest (RF) classifier to design pattern detection. By using BFA technique, the method used only five metrics for the identification of patterns, where existing techniques uses 70 to 80 metrics for same pattern detection. The simulations are conducted to test the effectiveness of BFA with ensemble classifiers on the Python Software platform in terms of average accuracy and precision. The results stated that BFA-RF achieved 88.57% of average accuracy, where BFA-SVM technique achieved 81.43% of average accuracy, which shows the RF achieved better results among other ensemble classifiers.

**Keywords:** Design patterns, Feature selection, Machine learning, Python software, Source code.

## 1. Introduction

A design pattern is defined as an abstract, repeatable solution to a typical software development problem in a specific context. The object-oriented design is typically known as partial design consisting of classes to define the measures and capabilities of objects [1]. The design patterns in software development provide several benefits, like increased reuse, modularity, quality, consistency between project and implementation, and the relationship between a development team and developers [2]. Basically, the design patterns are superior kind of software construction that deliver system information at a higher end of abstraction, while software design patterns offer information on a low end of abstraction [3]. The identification of design patterns from source code is an important task in the reverse engineering process and it provides huge benefits of understanding program code when documentation is inadequate or missing, provides project information to help restore the software architecture and enable verification of source code compliance and design [4]. The existing third-party programs and open source software may take a long time to understand for developers, because the models can be applied without explicit class names, comments or attachments. The study samples should improve the clarity of the program [5, 6] for its better understanding to the developers. However, manual sampling in existing programs is inefficient and sampling can be excluded. To overcome the aforementioned problems, the static characteristics of the samples are used in many studies on the identification of the samples [7, 8]. This static analysis makes it difficult to identify the patterns in which the structures are similar. Therefore, it is necessary to define the role of the candidate class and the models that help to understand and support the initial design decision of the application [9, 10]. In this study, the sample codes for pattern identification is implemented on the Python Software, where in the existing SVM, the decision tree uses the sample code on JAVA platform [11]. The reason for choosing Python is highly productive and interpreted language with elegant syntax than JAVA and it is a good option for rapid application development and scripting. Moreover, the existing techniques included all feature metrics nearly 70 metrics to detect patterns which leads high computation time. The research study uses the BFA as feature selection technique to solve the issues of existing techniques by selecting the important feature metrics based on their correlation towards the targeted design pattern type as the fitness function. While comparison with traditional techniques, the BFA with ensemble classifiers required only five feature metrics from the input source code for pattern identification. The proposed BFA is validated with ensemble classifiers namely Naïve Bayes (NB), SVM, RF and decision tree and also studied the importance of classes as a case study.

The organization of the research study is given as follows: Section 2 illustrates the study of various existing techniques for design pattern detection. The explanation of proposed BFA-RF is presented in Section 3. The simulations are conducted as a case study, quantitative analysis and comparative study of proposed feature selection are given in Section 4. The conclusion of the research study is explained in Section 5.

## 2. Literature review

This section presents the design patterns of source code in existing techniques with its key benefits and limitations;

Dwivedi [11] developed three classifiers namely Artificial Neural Network (ANN), RF and SVM to detect the software design patterns. The two phases such as developing the dataset based on metrics and identifying the design patterns were considered in the study. The design patterns such as bridge, adapter, template, abstract factory and composite were used in this study. The open software sources JUnit, JHotDraw and QuickUML were used to test the efficiency of the developed scheme and the results stated that it achieved better accuracy by reducing the total number of candidate classes. The main drawback of the study was that it does not provide the correct instances of patterns due to unavailability of standard benchmark.

Chihada et al. [12] designed a ML based model (i.e. Extension of SVM) to detect the design patterns. The various versions of design patterns were identified by the proposed ML technique by extracting the information from the instances of design patterns. The number of candidate design patterns was greatly reduced from a given source code by developing a novel method pre-processing technique. The developed method was applied on the open source codes to detect the six various design patterns in the simulations. The classification accuracy was minimized and computation time was also increased, while predicting metrics from the source code.

Zanoni [13] implemented a Metrics and ARchitecture Reconstruction PLugin for Eclipse (MARPLE) to solve the problem of design pattern identification. A pattern samples contained a variable number of classes that were represented by leveraging a design pattern modeling. The MARPLE method used five patterns such as Factory Method, Adapter, Singleton, Decorator and Composite on ten open source software systems. The validated results stated that MARPLE methods have significant performance on four patterns, but it showed lower performance on the Composite patterns.

Dwivedi [14] recognized the design patterns by implementing the ANN and logistic regression models. In the process of identification, the developed method used six various types of metrics on three open source platforms namely Quaqua, JRefactory and JUnit. The parameters' quality of the learning methods was improved by retrieving the data from the source code. The simulation results proved that the number of candidate patterns was minimized to achieve better recognition accuracy. However, the computation time of the supervised learning model was high due to the absence of optimization techniques.

Mhawish and Gupta [15] implemented a tree-based ML algorithm to detect the software metrics and design patterns. For each role, the metrics were calculated and extracted the design patterns roles to develop the dataset called P-MARt repository. The input parameter of tree-based algorithm was optimized by designing a Grid Search Algorithm (GSA). The redundant features were removed and the knowledge of software metrics was improved by introducing the Genetic Algorithm based NB and correlated features of the target class (CFS). The developed method used only two design patterns in simulations such as Adapter/ConcreteCommand and Adaptee/Receiver. This method failed to analyse the quality and presence of design patterns.

Uchiyama [16] proposed a ML algorithm and used the source code metrics for pattern detection

Received: July 18, 2020. Revised: November 5, 2020.

technique. According to the metrics, candidates were analysed by using the developed technique and the patterns were identified from the relations of classes. The simulations used the small scale and large-scale programs from the JAVA library as sample data to validate the performance of this method. But, the scheme had more false positives, because strict conditions were not used in this scheme.

Abd Manaf [17] implemented a hybrid Simple Linear Iterative Clustering (SLIC)-Extra Tree (ET) algorithm for object-based image analysis (OBIA) containing two phases, namely object-based classification and fast segmentation. In the first stage, three segmentation algorithms were used, namely Felzenszwalb (Felz), Quickshift (QS), and SLIC, to segment a Landsat image into superpixels. The second step involves applying object extraction to the images in the section to extract several important functions. Therefore, a series of experiments were conducted using 15 mL methods to classify the images in the section. The results proved that the classification of ETs in a combination of three segmentation algorithms Felz, QS and SLIC were the most accurate classifier for OBIA-based image segmentation, achieving maximum total accuracy. The results proved that SLIC produced compact superpixels which effectively minimized the computational overhead of the classification process.

Huang [18] developed a Feature Clustering SVM with Recursive Feature Elimination (FCSVM-RFE) to enhance SVM-RFE for gene selection by incorporating the K-means clustering method. The FCSVM-RFE has three phases: gene clustering, gene presentation and gene sequencing and it used to reduce the complexity and redundancy among the genotypes. The gene clustering was implemented using K-mean clustering to find representative genes for genetic clusters. The SVM-RFE has been used to sequence representative genes. The simulation results showed that FCSVM-RFE achieved better classification performance and less computational complexity than the other existing methods. The FCSVM-RFE requires high run time on Breast Cancer dataset than other datasets.

From the analysis of existing techniques, it is clearly stated that optimization technique is required to select the important features and these existing techniques used only JAVA software platform for the design pattern detection. To overcome aforementioned issues, this research study implements the BFA with RF to choose the relevant features and increase the classification accuracy. The proposed BFA is validated in terms of average accuracy, precision, recall and F-measure, the other feature selections namely ET classifier [17] and RFE [18] are and these algorithms are also implemented on the collected metrics used in research study.

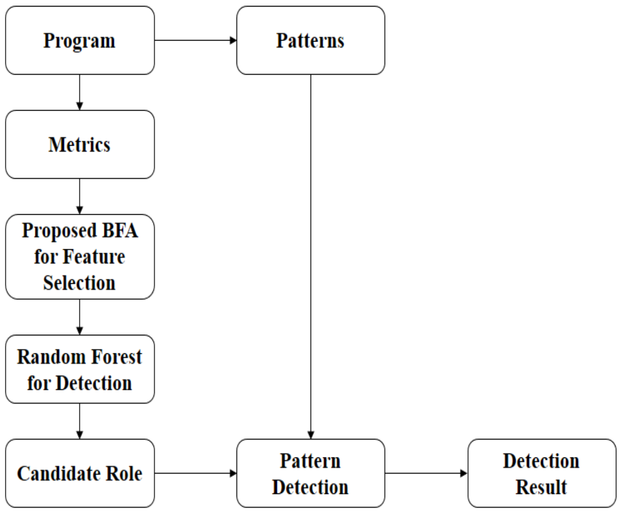

Figure.1 Working Procedure of Proposed Methodology

The proposed BFA method uses the Python software platform to identify design patterns, which is explained in following section.

## 3. Proposed methodology

An object-oriented program plays a major role in improving the understandability, maintainability, and reusability through the detection of patterns. Many traditional methods rely on static study to detect the pattern instances and use rigorous situations based on structure class data. It is complex to identify and differentiate the scheme patterns in which class arrangements are parallel. To overcome this problem, a method of recognizing the design model that optimizes the metrics of the source code using BFA and then, a ML algorithm called RF predicts the design patterns from the source code. The working procedure of the proposed method is presented in Fig. 1.

### 3.1 Creation of dataset

In the first step, a metrics-based data set is created to train the classifiers based on the training and testing samples used in this study. The seven types of patterns namely Singleton, Adapter, Facade, Factory, Proxy, State and Template patterns are selected to create a dataset. The concept of a design model is often defined in terms of many aspects such as problem, consequences, motivation, structure and behaviour. To extract the patterns instance, source code from the selected software (Python) is used as input to extract pattern instances.

Received: July 18, 2020. Revised: November 5, 2020.

## 3.2 Feature selection using bacterial foraging algorithm

Once the dataset is developed, the important features 'print_cnt', 'Class_cnt', 'condition_cnt', 'loop_cnt', 'def_counts', 'import_cnt', 'comment lines', 'code lines', 'blank lines', 'lines', are given as input to BFA that leads to effectively improve the classification accuracy of ensemble classifier. The BFA is a bio-inspired global optimization mostly taken in the control problem then the distributed optimization. The BFA is based on the intellectual behaviour of society and collaboration in Escherichia coli bacteria.

It represents the activities of communal type bacteria Escherichia coli bacteria lives in the human gut. The individual bacterial movement of Escherichia coli is supported by a series of pulling flagella. The Escherichia coli alternated two main bacterial operations running and falling in the human intestines during feeding [19]. The aim of the optimization strategy used in BFA is to apply a distorted random shift for every one bacterium and it increa ses the concentration of nutrients, avoid harmful substances and immediately leave the neutral environment. During this behaviour, the four main operations are considered while bacterial feeding, i.e. chemotaxis, mutation of tumble and swarms process, reproduction and elimination of variance [20]. The flow chart for working process of BFA is presented in Fig. 2 [22].

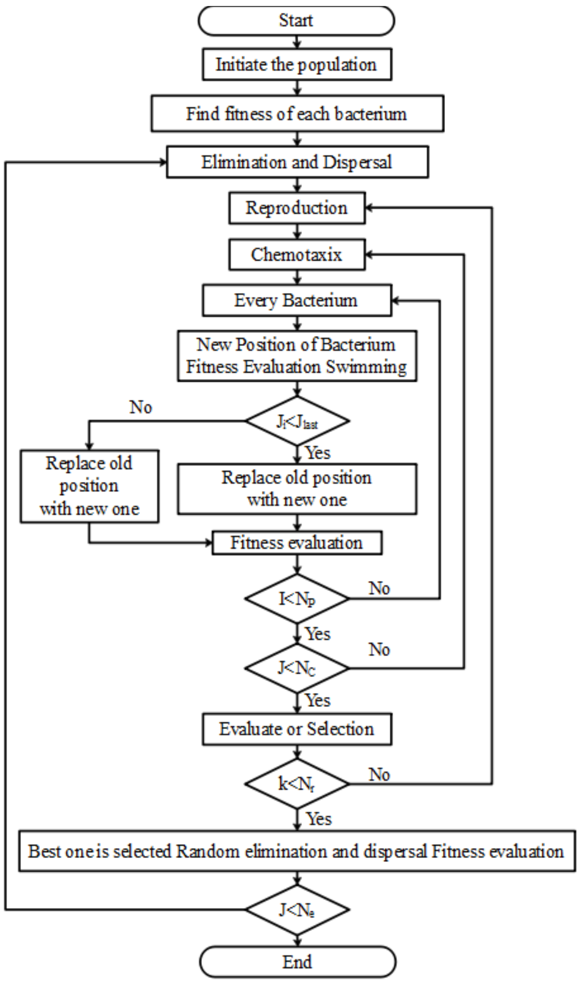
Start
Initiate the population
Find fitness of each bacterium
Elimination and Dispersal
Reproduction
Chemotaxix
Every Bacterium
New Position of Bacterium Fitness Evaluation Swimming
No
Ji<Jlast
Yes
Replace old position with new one
Replace old position with new one
Fitness evaluation
I<Np
No
Yes
J<Nc
No
Yes
Evaluate or Selection
k<Nr
No
Yes
Best one is selected Random elimination and dispersal Fitness evaluation
J<Ne
End

Figure. 2 Working procedure of BFA

### 3.2.1. Chemotaxis

Each and every individual in the BFA are simulated by Chemotaxis process. Chemotaxis is the movement of a moving cell or portion of it in a way that corresponds to the gradient of growing or falling concentrations of nutrients or toxic matters. The steps of chemotaxis are communicated to a tumble followed by a run or a tumble followed by a tumble. The following Eq. (1) represents the case.

$$\Phi(i) = \left(\Delta^T(i)\Delta(i)\right)^{-1/2} \times \Delta(i) \qquad (1)$$

Where $\Delta(i) = \{\Delta_1(i),\ \Delta_2(i) \dots, \Delta_p(i)\} \in \mathbb{R}^p$ is a randomly engendered vector living in $M$ and having at smallest unit norm, whose every element $\Delta_m(i), m = 1,2, \dots., p$ is a randomly taken in $[-1,1]$. The experimental chemotaxis is expressed in the Eq. (2) based on the movement of the bacterium.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i) \times \Phi(i) \quad (2)$$

Where, $i, j, k$ $and$ $l$ are the parameter of BFA. $C(i)$ represents the size of the steps taken in the random direction that is specified by the tumble.

### 3.2.2. Swarming

It is a group of bacteria that organize themselves. The group was monitored for mobile bacteria, including Escherichia coli, to achieve self-organizing activity. The Cell-cell signalling can be achieved through attractiveness and a diffusion pattern between bacteria. The purpose of this process is to inform each other about nearby nutrients and pollution. Cell-cell signalling is modeled by the next Eq. (3).

$$J_{CC}\left(\theta, \theta^i(j,k,l)\right) = -da \times exp\left(-w_a \times \sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right) + h_r \times exp\left(-w_r \times \sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right) \qquad (3)$$

Where $d_a$ is the attractant's depth which is released by the cell, $w_a$ is the attractant signal's width, $h_r = d_a$ is the repellant effect height, and $w_r$

is the repellant width. The objective function is to be added to the actual objective function to acquire a time changing objective function which is given in the following Eq. (4).

$$J_{CC}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{CC}\left(\theta, \theta^i(j,k,l)\right) \quad (4)$$

Where, $P(j,k,l) = \{\theta^i(j,k,l), i = 1,2, \dots S\}$ represents the sum of optimized variable. $\theta^i(j,k,l)$ is the position of every separated bacteria in the population of the $S$ bacteria at the $j^{th}$ chemotactic phase, $k^{th}$ reproduction phase and $l^{th}$ elimination–dispersal result. The actual fitness function $J(i, j + 1, k, l)$ is calculated by adding the last value $J_{last}$ of $J(i,j,k,l)$, the cell-to-cell attractant–repellant profile to pretend the swarming manners. The fitness function is computed in Eq. (5) and the rate of $J(i,j+1,k,l)$ is computed in the following Eq. (6) [19].

$$J(i,j,k,l) = J(i,j,k,l) + J_{CC}\left(\theta^i(j,k,l), P(j,k,l)\right) \quad (5)$$

$$J(i,j+1,k,l) = J(i,j+1,k,l) + J_{CC}\left(\theta^i(j+1,k,l), P(j+1,k,l)\right) \quad (6)$$

Where, $\theta^i(j+1,k,l)$ is the cost at $J(i,j+1,k,l)$ is better when compared to $\theta^i(j,k,l)$. By using the Eq. (5 and 6), the important features are obtained.

### 3.2.3. Reproduction:

The Reproduction procedure is established on bacterial health resulting from the superiority of the food. The healthy or unhealthy bacteria die, where healthy bacteria provide excellent value for the target activity and can be differentiated into two bacteria located in the same location. This course retains the size of Escherichia coli constant. The $S_r$ bacteria with the maximum $J_{health}$ values die and the residual $S_r$ bacteria share the greatest values. The BFA does this by assigning copies made in the similar location as their parents. $N_{re}$ repeats the play step once as shown in Eq. (7).

$$J_{health}^{i} = \sum_{j=1}^{N_{C+1}} J(i,j,k,l) \quad (7)$$

Where $N_c$ is the sum of chemotactic stages.

### 3.2.4. Dispersal and elimination:

Escherichia coli bacteria show quick variations in the human intestines for several reasons. The number of people with high nutritional gradients is decreasing due to an increase in local temperature and this occurrence is termed as elimination course. The elimination and diffusion are the result of chemotaxis support, as bacteria are dispersed near good food sources. Some bacteria are killed by the minor probability $P_{ed}$, while the novel substitutes are set over the search space. Here, the number of population is 50 and initialize the space dimension as 3. At first, the elimination-dispersal process will be carried out to select only the nutrient bacteria for the reproduction process. In this study, the probability of the elimination-dispersal process is 1.15, if the population is less than 1.15, it will be eliminated from the group. If the population is higher than this probability, it will assist chemotaxis the movement, according to Eq. (2). E. coli moves from one place to another via flagella in this chemotaxis step. Its motions are considered in two different ways according to its biological view, where two ways includes either swim or tumble. In this research study, the swimming length is set to 12 and tumble value is set to 0.2, where chemotactic steps is set to 2. Identify the better cost by Eq. (5) and save this as $J_{last}$ and actual fitness function is identified by using Eq. (6). If the element value m is less than swimming length, add $m = m + 1$ and if the actual fitness value is less than $J_{last}$, then Eq. (2) is used to compute the new actual fitness function or else set $m = 12$. If $j$ is less than chemotactic, this process will be continued, since the life of the bacteria is not over. In order to identify the healthy bacteria, Eq. (7) is used and the total iteration of the algorithm is set to 100. By repeating the process, the best fitness value is stored and the output will be optimal feature metrics. The important features 'def_counts', 'object_counts', 'import_cnt', 'comment lines' and 'code lines', are given as an input to the ensemble classifiers, which is studied in following section.

## 3.3 Classification using ensemble classifier

After achieving the optimal feature values from BFA, ensemble classifier is implemented in this research study to classify the design patterns. A suitable kernel in SVM is selected to solve the overfitting problems, where those kernel parameters are fine-tuned and applied it to the k-fold cross validation. The main advantage of SVM is that it automatically selects its model size. However, the number of features used in SVM is small, therefore, model complexity is increased in the SVM. In the decision tree, nodes are splits according to the rules of splitting for every specific feature. From root to leaves, the data are passed in the decision tree and values of features are separated according to splitting rules and it predicts the class, when the stopping criteria reaches. The hyper parameters of maximal depth and split criterion must be optimized using the techniques of parameter optimization.

The RF is a superior classification technique to train a large dataset, because it uses only the low number of feature values to classify the patterns [21]. Additionally, the undertaken classification technique is a non-parametric pattern technique that significantly diminishes the issue of density of continuous random variables or probability density complexity. In RF, each tree is assumed as a distinct classifier, which are used to achieve better decision making and the growth rules of each tree is examined to develop a robust RF classifier.

The RF is one of the best ensemble classification technique that works based on the principle of bagging and uses decision tree as a base classification technique. Initially, extracted feature vectors are randomly sampled for training sets $N$. Next, select the sub feature values from the extracted feature vectors if $m(m < M)$ , where $M$ is indicated as extracted feature vectors. At last, selects $m$ feature vectors from the $M$ feature values and then split the nodes using best spilt on the $m$ dimensional feature vectors.

The error rate of RF classifier's depends on two factors:

- Tree strength needs to be high in order to diminish the error rate.
- The error rate can be reduced by lowering the correlation among the trees.

**Pseudo code of random forest**

```
Input:
Number of trees in random forest "n"
Number of feature vectors "M"
Training samples "N"
Proportion of feature vectors considered to build tree "m"
Output:
        $Ensemble$ "$m$"
    1. $E \leftarrow 0$
    2. For $i = 1$ to $n$ do
    3. $N^i \rightarrow Bootstrap\ sample\ (N)$
    4. $A^i \rightarrow Random\ select\ (m)$
    5. $C^i \rightarrow Build\ random\ forest\ (N^i, A^i)$
    6. $EU\{C^i\} \rightarrow E$
    7. End for
    8. Return $E$
    9. End algorithm
```

The overfitting is one of the critical problems that may affect the results worse, but for RF algorithm, if there are enough trees in the forest, the classifier won't overfit the model. The main advantages RF classifier are it can handle the missing values and modelled the categorical values compared to other ML algorithms, namely SVM and Decision Tree. The final output of RF has Role agreement value, Relation agreement value, Pattern agreement value to identify the candidate role. These three values are defined as follows:

- **Role agreement value:** The values of a rolling contract above a threshold considered in the candidate roles.
- **Relation agreement value**: The relationship agreement was introduced to understand the variance between interface implementation, inheritance, and aggregation relationships.
- **Pattern agreement value:** The relation agreement and the role agreement value are used to gain the pattern agreement value.

These candidate values are combined with the patterns and finally identify the design patterns as detection outputs namely creational, behavioural and structural patterns. That is, a singleton of creativity models, an adapter of structure models and a template model of behaviour models. The results and discussion of proposed BFA-RF with other ensemble techniques given as follows.

## 4. Results and discussion

The validation of proposed BFA-RF technique is carried out with other existing techniques namely SVM, decision tree and NB in terms of accuracy, precision, recall and F-score. The system with Intel i5 processor, 8GB RAM with 500GB hard disk is used to implement the proposed BFA-RF. The results section consists of case study, quantitative analysis of proposed BFA-RF technique and comparative study of BFA-RF with other ensemble techniques.

### 4.1 Parameter evaluation

The detection of software design patterns is measured with accuracy, recall, precision and F-measures (i.e., Harmonic average of precision and recall), as shown in the Eq. (8) to (11), respectively, where "NDP" in these equations indicates the number of instances of the patterns. A true positive result (TP)

Table 1. Class description used in the research study

| Class Number | Description of Class |
|---|---|
| 1 | Adapter |
| 2 | Facade |
| 3 | Factory |
| 4 | Proxy |
| 5 | Singleton |
| 6 | State |
| 7 | Template |

determines the patterns available in the dataset based on metrics that the classifier correctly determined. The False Positive (FP) identifies instances of patterns that does not exist in the metric-based dataset, and the classifier correctly identified them. The False Negative (FN) defines instances of the pattern in a dataset based on metrics that the classifiers do not identify correctly. The high value of the F-score indicates greater accuracy in the design pattern determination process.

$$Accuracy = \frac{\sum_{i=1}^{NDP} TP_i + TN_i}{\sum_{i=1}^{NDP} TP_i + TN_i + FP_i + FN_i} \tag{8}$$

$$Precision = \frac{\sum_{i=1}^{NDP} TP_i}{\sum_{i=1}^{NDP} TP_i + FP_i} \tag{9}$$

$$Recall = \frac{\sum_{i=1}^{NDP} TP_i}{\sum_{i=1}^{NDP} TP_i + FN_i} \tag{10}$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{11}$$

Where, True Negative is defined as TN.

## 4.2 Case study

The research study consists of seven classes as input to validate the performance of proposed BFA-RF, which are described in Table 1. Two different kinds of case studies: Adapter and Singleton are discussed in this section. The research study selected only two classes as case study, this is because more number of instances are presented in Adapter as Case Study 1 and optimized (i.e. less) instances are presented in Singleton as Case Study 2.

### 4.2.1. Case study 1

The example implementation of Adapter class in Python software is given as follows:

```
/* This is our Adapter, a third party implementation
of
# ----------------
# This is our Adaptee Class
# --------------
class Adapter:
    def adaptee_request(self):
        print("Adapter function called.")
# ----------------
# This is our Target Interface
# ----------------
class Target(ABC):
    """
        Interface for Client
    """
    def __init__(self):
        self._adapter = Adapter()
    @abstractmethod
    def request(self):
        pass
# ----------------
# This is our Adapter Class
# ----------------
class Adapter(Target):
    def request(self):
        self._adapter.adapter_request()
adapter = Adapter()
adapter.request()
```

The sample code shows Adapter class that has Adapter class, target interface and Adapter class. In the class Adapter, at least one method inherited from Target should call a method inherited from Adaptee. In the object Adapter, the field referencing the Adaptee may not be changed before its usage by the Target method overridden implementation: in other words, the state of the Adaptee field should be immutable. Therefore, this class consists of more number of instances than other classes. Moreover, Adapter contains three roles in a single (root) level and singleton described the patterns in one role, where all the remaining patterns are multi-level, i.e. they have at least one child level below the root one.

### 4.2.2. Case study 2

The case study-2 shows the sample code for Singleton class, which is implemented in Python software.

```
class Singleton(object):
     def __new__(cls):
        if not hasattr(cls, 'instance'):
            cls.instance = super(Singleton,
cls).__new__(cls)
        return cls.instance
```

Table 2. Performance analysis of adapter (case 1) and singleton (case 2) for various classifiers

| **Classifiers with BFA** | **Adapter** | | **Singleton** | |
|---|---|---|---|---|
| | **Precision (%)** | **F-Measure (%)** | **Precision (%)** | **F-Measure (%)** |
| Decision Tree | 100 | 100 | 73 | 84 |
| SVM | 97 | 99 | 70 | 78 |
| NB | 97 | 99 | 62 | 76 |
| Proposed RF | 100 | 100 | 73 | 84 |

Table 3. Performance analysis of proposed BFA-RF in terms of precision (%) for seven classes

| **Classifiers** | **Without Feature Selection** | | | | | | | **Proposed BFA** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Decision Tree | 40 | 46 | 55 | 50 | 46 | 48 | 47 | 73 | 96 | 47 | 68 | 93 | 80 | 100 |
| SVM | 38 | 36 | 40 | 36 | 35 | 43 | 41 | 70 | 93 | 45 | 79 | 100 | 58 | 97 |
| NB | 42 | 46 | 47 | 28 | 60 | 39 | 55 | 62 | 96 | 50 | 92 | 87 | 88 | 97 |
| Proposed RF | 41 | 43 | 55 | 50 | 49 | 53 | 50 | 73 | 100 | 50 | 83 | 100 | 89 | 100 |

Table 4. Analysis of different classifiers with BFA by means of recall (%)

| **Classifiers** | **Without Feature Selection** | | | | | | | **Proposed BFA** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Decision Tree | 42 | 46 | 52 | 49 | 47 | 46 | 39 | 100 | 79 | 100 | 65 | 59 | 100 | 100 |
| SVM | 37 | 35 | 43 | 38 | 35 | 43 | 39 | 88 | 79 | 100 | 65 | 59 | 88 | 100 |
| NB | 40 | 45 | 48 | 30 | 55 | 44 | 45 | 100 | 79 | 100 | 52 | 91 | 88 | 100 |
| Proposed RF | 46 | 45 | 50 | 55 | 47 | 48 | 45 | 100 | 88 | 100 | 65 | 82 | 100 | 100 |

Table 5. Performance of various classifiers with and without BFA on the basis of F-measure (%)

| **Classifiers** | **Without Feature Selection** | | | | | | | **Proposed BFA** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Decision Tree | 41 | 42 | 50 | 49 | 45 | 48 | 44 | 84 | 87 | 64 | 67 | 72 | 89 | 100 |
| SVM | 34 | 38 | 45 | 31 | 39 | 45 | 40 | 78 | 85 | 62 | 71 | 74 | 70 | 99 |
| NB | 41 | 43 | 45 | 32 | 58 | 46 | 43 | 76 | 87 | 67 | 67 | 89 | 88 | 99 |
| Proposed RF | 43 | 46 | 51 | 57 | 46 | 44 | 49 | 84 | 94 | 67 | 73 | 90 | 94 | 100 |

```
s1 = Singleton()
```

This sample code shows that the Singleton has only one instance (i.e. optimized class) called object instance. The performance of different classifiers namely SVM, NB, RF and decision tree on only Adapter and Singleton in terms of Precision and F-Measure is illustrated in Table 2. The feature selection BFA is implemented with SVM, NB, RF classifiers and shows the validated results.

The validated results shown in Table 2 prove that the proposed RF with BFA achieved better performance, even when the data samples has more number of instances. The RF achieved 100% of precision and recall, where SVM and NB achieved 97% of precision and 99% of F-measure on adapter class for more number of instances. The validation on optimized instances is low, when compared with more number of instances.

The proposed RF achieved 73% of precision and 84% of F-measure, where the decision tree achieved only 73% of precision and 84% of F-measure for the singleton class instance. This proves the efficiency of proposed RF with BFA will not affected with more number of instances.

## 4.3 Quantitative analysis of proposed BFA-RF technique

In this section, the performance of proposed BFA with RF technique is compared with other existing classifiers namely SVM, decision tree and NB on seven class metrics in terms of precision, recall and F-measure. The comparative study of three classes' adapter, singleton and Factory of RF with other classifiers is also discussed in the following section.

### 4.3.1. Performance of proposed BFA-RF in terms of precision

Table 3 describes the analysis of precision on seven classes for the different classifiers with and without feature selection technique. In the following

Table 3 to 5, the numbers 1 to 7 represents the class numbers.

The different classifiers without feature selection technique provides poor precision for all classes. For instance, SVM, NB and RF achieved 43%, 39% and 53% of precision for the class 6. This is because, all the collected features metrics are given as input for design pattern prediction, which leads high computation time. In order to address these issues, the feature selection technique called BFA is introduced in the research study and experiments is conducted on every class. Therefore, the existing SVM, NB and RF with BFA achieved 58%, 88% and 89% of precision for class 6. This shows that the proposed BFA-RF technique achieved higher performance in terms of precision for all seven classes. Table 4 shows the performance analysis of various classifiers with BFA on the basis of Recall for seven classes.

The existing decision tree, SVM and NB with BFA achieved only 79%, 79% and 79% of recall, where proposed RF with BFA achieved 88% of recall for the class 2. This is because, RF handles the missing values and categorical data, where other classifiers are insufficient to handle those data. The RF effectively identifies the design patterns from the source code and RFA without BFA achieved 55% of recall, where Decision tree, SVM and NB without BFA achieved 49%, 38% and 30% for the class 4 respectively. Table 5 presents the performance of classifiers with and without feature selection technique by means of F-measure on all seven classes.

By implementing the BFA in different classifiers, the performance of these classifiers is increased in terms of F-measure for seven classes. The results of F- measure proves that the importance of selecting relevant metrics from the input data that leads higher performance of various classifiers, for example, decision tree, SVM, NB and RF with BFA achieved 87%, 85%, 87% and 94% of F-measure, where the classifiers without BFA achieved only 42%, 38%, 43% and 46% for the class 2 respectively.

## 4.4 Comparative study of proposed feature selection technique (BFA)

In this section, the performance of proposed BFA-RF is compared with other existing feature selection techniques namely ET classifier [17] and RFE [18]. The other classifiers namely SVM, decision tree and NB are also implemented with these feature selection techniques on the collected dataset in terms of average accuracy, which is shown in Table 6.

From the Table 6, the validated results showed that the proposed feature selection BFA achieved better performance with the RF classifier than other classifier techniques. The existing ET and RFE feature selection techniques are implemented in this research study with other classifiers even when the proposed RF classifier achieved low average accuracy 48.66% without feature selection techniques. When implementing the ET and RFE with RF, the average accuracy is nearly 58% to 59%, this is because the source code samples have more important features and it’s not effectively selected by the ET and RFE eliminates many code samples from the data, which leads poor accuracy. Table 7 shows the comparative analysis of BFA, ET and RFE on the basis of average precision.

The existing SVM classifier is implemented with ET, RFE and proposed BFA and achieved the average precision as 55% and 57% for existing ET,

Table 6. Comparative study of proposed BFA-RF in terms of average accuracy (%)

| Classifier | Without Feature selection | ET [17] | RFE [18] | Proposed BFA |
|---|---|---|---|---|
| Decision Tree | 46.33 | 54.77 | 56.32 | 82.86 |
| SVM | 37.36 | 47.44 | 51.52 | 81.43 |
| NB | 44.88 | 54.16 | 56.88 | 85.00 |
| Proposed RF | 48.66 | 58.52 | 59.98 | 88.57 |

Table 7. Comparative study of BFA-RF on the basis of average precision (%)

| Classifier | Without Feature selection | ET [17] | RFE [18] | Proposed BFA |
|---|---|---|---|---|
| Decision Tree | 47.00 | 55.00 | 57.00 | 87.00 |
| SVM | 38.00 | 48.00 | 53.00 | 87.00 |
| NB | 45.00 | 55.00 | 57.00 | 89.00 |
| Proposed RF | 49.00 | 59.00 | 60.00 | 92.00 |

Table 8. Comparative analysis of proposed BFA with various classifiers in terms of average recall (%)

| Classifier | Without Feature selection | ET [17] | RFE [18] | Proposed BFA |
|---|---|---|---|---|
| Decision Tree | 46.00 | 54.00 | 56.00 | 83.00 |
| SVM | 38.00 | 47.00 | 53.00 | 81.00 |
| NB | 44.00 | 54.00 | 56.00 | 85.00 |
| Proposed RF | 48.00 | 58.00 | 59.00 | 89.00 |

Table 9. Comparative analysis of proposed BFA with different classifiers on the basis of average F-Measure (%)

| Classifier | Without Feature selection | ET [17] | RFE [18] | Proposed BFA |
|---|---|---|---|---|
| Decision Tree | 46.00 | 54.00 | 56.00 | 83.00 |
| SVM | 38.00 | 47.00 | 53.00 | 82.00 |
| NB | 44.00 | 54.00 | 56.00 | 85.00 |
| Proposed RF | 48.00 | 58.00 | 59.00 | 89.00 |

RFE, but SVM with BFA achieved 87% of average precision. Similar to SVM technique, the Decision tree with BFA achieved higher 87% of average precision than ET and RFE techniques. The RF with BFA is achieved 92% of average precision which is high when compare to other classifiers namely decision tree, SVM and NB. The proposed RF handles the missing values, where other classifiers are insufficient to handle those missing values or instances. Table 9 presented the comparative study of different features selections with various classifiers by means of average recall.

## 5. Conclusion

In this study, an effective feature selection (BFA) technique with RF classifier is proposed to detect the design patterns from the sample input codes. The existing techniques namely decision tree, NB and SVM are not incorporated with optimization technique to detect the patterns that lead less accuracy. The research study developed the BFA to overcome aforementioned issues and choose the relevant metrics from the seven classes. The optimal features were given as an input to ensemble classifiers for final pattern detection. The simulated codes of this research were implemented in the Python software, where the existing techniques were implemented in JAVA software. Therefore, the decision tree, SVM and NB are also implemented on the simulated codes of this research study. The validated results proved that the proposed BFA-RF achieved 88.57% of average accuracy, 92% of average precision, 89% of average recall and 89% of average F-measure, where Decision tree with BFA achieved 82.86% of average accuracy, 87% of average precision, 83% of average recall and 83% of average F-measure. The experimental results proved that the BFA-RF achieved 100% of precision, where RF without BFA achieved only 43% on Facade class. At last, a case study with more number of instances and less number of instances were also discussed in the experimental section. In the future work, the proposed BFA-RF extends the process of pattern detection in various open-source software to improve the accuracy values.

## Conflicts of Interest

The authors declare no conflict of interest.

## Author Contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing-original draft preparation, writing-review and editing, visualization, have been done by 1st author. The supervision and project administration, have been done by 2nd author.

## References

[1] S. M. H. Hasheminejad and S. Jalili, "Design patterns selection: An automatic two-phase method", *Journal of Systems and Software*, Vol. 85, No. 2, pp. 408-424, 2012.

[2] F. A. Fontana and M. Zanoni, "A tool for design pattern detection and software architecture reconstruction", *Information Sciences*, Vol. 181, No. 7, pp. 1306-1324, 2011.

[3] A. Wierda, E. Dortmans, and L. Somers, "Pattern detection in object-oriented source code", *Software and Data Technologies, Springer*, Berlin, Heidelberg, pp. 141-158, 2007.

[4] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidis, "Design pattern detection using similarity scoring", *IEEE Transactions on Software Engineering*, Vol. 32, No. 11, pp. 896-909, 2006.

[5] A. K. Dwivedi and S. K. Rath, "Formalization of web security patterns", *INFOCOMP Journal of Computer Science*, Vol. 14, No. 1, pp.14-25, 2015.

[6] B. B. Mayvan and A. Rasoolzadegan, "Design pattern detection based on the graph theory", *Knowledge-Based Systems*, Vol. 120, pp. 211-225, 2017.

[7] H. Zhu and I. Bayley, "On the composability of design patterns", *IEEE Transactions on Software Engineering*, Vol. 41, No. 11, pp. 1138-1152, 2015.

[8] A. K. Dwivedi and S. K. Rath, "Incorporating security features in service-oriented architecture using security patterns", *ACM SIGSOFT Software Engineering Notes*, Vol. 40, No. 1, pp. 1-6, 2015.

[9] G. Rasool and P. Mäder, "A customizable approach to design patterns recognition based on feature types", *Arabian Journal for Science and*

*Engineering*, Vol. 39, No. 12, pp. 8851-8873, 2014.

[10] I. N. IssaouiBouassida and H. Ben-Abdallah, “Using metric-based filtering to improve design pattern detection approaches”, *Innovations in Systems and Software Engineering*, Vol. 11, No. 1, pp. 39-53, 2015.

[11] A. K. Dwivedi, A. Tirkey, and S. K. Rath “Software design pattern mining using classification-based techniques”, *Frontiers of Computer Science*, Vol. 12, No. 5, pp. 908-922, 2018.

[12] A. Chihada, S. Jalili, S. M. H. Hasheminejad, and M. H. Zangooei, “Source code and design conformance, design pattern detection from source code by classification approach”, *Applied Soft Computing*, Vol. 26, pp. 357-367, 2015.

[13] M. Zanoni, F. A. Fontana, and F. Stella, “On applying machine learning techniques for design pattern detection”, *Journal of Systems and Software*, Vol. 103, pp. 102-117, 2015.

[14] A. K. Dwivedi, A. Tirkeyand, S. K. Rath, “Applying learning-based methods for recognizing design patterns”, *Innovations in Systems and Software Engineering*, Vol. 15, No. 2, pp. 87-100, 2019.

[15] M. Y. Mhawish and M. Gupta “Software Metrics and tree-based machine learning algorithms for distinguishing and detecting similar structure design patterns”, *SN Applied Sciences*, Vol. 2, No. 1, pp. 11, 2020.

[16] S. Uchiyama, A. Kubo, H. Washizaki, and Y Fukazawa, “Detecting design patterns in object-oriented program source code by using metrics and machine learning”, *Journal of Software Engineering and Applications*, Vol. 7, No. 12, pp. 983, 2014.

[17] S. Abd Manaf, N. Mustapha, M. N. Sulaiman, N. A. Husin, H. Z. M. Shafri, and M. N. Razali, “Hybridization of SLIC and Extra Tree for Object Based Image Analysis in Extracting Shoreline from Medium Resolution Satellite Images”, *International Journal of Intelligent Engineering and Systems*, Vol. 11, No. 1, pp. 62-72, 2018.

[18] X. Huang, L. Zhang, B. Wang, F. Li, and Z. Zhang, “Feature clustering-based support vector machine recursive feature elimination for gene selection”, *Applied Intelligence*, Vol. 48, No. 3, pp. 594-607, 2018.

[19] K. M. Passino, “Biomimicry of bacterial foraging for distributed optimization and control”, *IEEE Control Systems Magazine*, Vol. 22, No. 3, pp. 52-67, 2002.

[20] W. N. E. A. W. Afandie, T. K. A. Rahman, and Z. Zakaria, “Comparative Analysis of Bacterial Foraging Optimization Algorithm and Evolutionary Programming for Load Shedding in Power System”, *International Journal of Simulation--Systems, Science & Technology*, Vol. 17, No. 41, 2016.

[21] B. Xu, X. Guo, Y. Ye, and J. Cheng, “An Improved Random Forest Classifier for Text Categorization”, *JCP*, Vol. 7, No. 12, pp. 2913-2920, 2012.

[22] I. Ullah, Z. Khitab, M. N. Khan, and S. Hussain, “An efficient energy management in office using bio-inspired energy optimization algorithms”, *Processes*, Vol. 7, No. 3, pp. 142, 2019.