IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 November 2023, accepted 4 December 2023, date of publication 8 December 2023,
date of current version 20 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3341395

RESEARCH ARTICLE

Meta-Heuristic Guided Feature Optimization for Enhanced Authorship Attribution in Java Source Code

BILAL AL-AHMAD[1,2], NAILAH AL-MADI[3], ABDULLAH ALZAQEBAH[4], RAMI S. ALKHAWALDEH[1], KHALED ALDEBEI[1], MD. FAISAL KABIR[5], (Member, IEEE), ISMAIL ALTAHARWA[1], MUA'AD ABU-FARAJ[1], AND IBRAHIM ALJARAH[6], (Senior Member, IEEE)

[1]The University of Jordan, Aqaba 77110, Jordan
[2]Department of Computer Science and Information Technology, Saint Cloud State University, St. Cloud, MN 56301, USA
[3]Princess Sumaya University for Technology, Amman 11941, Jordan
[4]Computer Science Department, Faculty of Information Technology, Al-Ahliyya Amman University, Amman 19628, Jordan
[5]School of Science, Engineering, and Technology, Pennsylvania State University Harrisburg, Middletown, PA 17057, USA
[6]The University of Jordan, Amman 11942, Jordan

Corresponding author: Ibrahim Aljarah (i.aljarah@ju.edu.jo)

ABSTRACT Source code authorship attribution is the task of identifying who develops the code based on learning based on the programmer style. It is one of the critical activities which used extensively in different aspects such as computer security, computer law, and plagiarism. This paper attempts to investigate source code authorship attribution by capturing natural language aspects of the code rather than only using minimal set of syntactic and stylistic code features as explored in the previous literature. It proposes an evolutionary feature selection model to improve the accuracy of authorship attribution by implementing two language models (uni-gram and bi-gram). The proposed approach uses K-Nearest Neighbor as a classifier and Genetic Algorithm as a feature selection technique. Two experiments have been demonstrated on a public Authorship Attribution dataset on GitHub, the experiments include various evolutionary feature selection models. Notably, the obtained results in both experiments were compared with the related studies, and show a significant improvement in terms of accuracy.

INDEX TERMS Evolutionary computation, data mining, feature selection, java, source code, authorship attribution.

I. INTRODUCTION

Authorship attribution is one of the challenging tasks explored in the field of Natural Language Processing (NLP). It aims to identify the author of a given source code among several known authors [1]. Source code authorship attribution is the task of deciding who wrote the code by identifying its attributes and styles. It is used in various problems such as code plagiarism, software forensics, academia, and copyright. Several methods of source code authorship attribution have been proposed as in the studies [2], [3], [4], [5], [6], [7].

In recent times, the issue of software plagiarism has emerged as a significant concern, and the identification of the source code's authorship can play a crucial role in detection. The process of source code authorship attribution involves determining the original author of a given source code from a known group of authors [1]. Apart from combating software plagiarism, this attribution technique also holds practical value in resolving authorship disputes, conducting software forensics, and tracing malicious code code [8], [9], [10]. Consequently, the endeavor to identify source code authors

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Tucci.

based on their unique stylistic features has gained significant attention.

The SCAP (Source Code Author Profiles) technique was developed to analyze the authorship of source code [2]. Another version of the SCAP approach was explored for author identification purposes [4]. The study utilized Java and C++ programs for its investigation. Self-organizing maps (SOM) have also been employed to identify the unknown author's identity [3]. Research has also explored the enhancement of code authorship identification through integrating feature sets and applying information retrieval ranking techniques or machine classifiers [6].

Moreover, numerous research studies have concentrated on investigating the identification of authors based on programming code written in the Java language. Noteworthy studies [6], [11], [12], [13], [14] have been conducted in this domain. The authors in the paper [11] used Krsul's C metrics for Java and utilized byte-level n-gram profiles to measure their approach. Also, the authors in the paper [12] used 18 metrics as histogram distributions from source codes. The study used these metrics and genetic algorithms (GA) to identify authors in software forensics. Another study narrowed down the metrics from eighteen to only four and used a genetic algorithm to discriminate them [13].

The previous research used a few sets of predetermined syntactic and stylistic features of Java. Whereas, this paper aims to deeply explore the Java source code features from a natural language perspective by covering large number of source code features. It mainly uses a K-Nearest Neighbor (KNN) classifier and GA as an evolutionary feature selection to improve the accuracy of authorship attribution by considering natural language features that would help to accurately distinguish the programmers. To validate the effectiveness of the proposed approach, experiments were conducted on the Java GitHub dataset.[1]

The rest of the paper is structured as follows. Section II examines the literature review. Section III outlines the research methodology employed. The analysis of results and discussion is presented in Section IV. Section V delves into the summary of the research findings. Limitations of this work are identified in Section VI. Finally, Section VII concludes the paper and outlines directions for future work.

II. LITERATURE REVIEW

A. BACKGROUND

Multiple NLP and classification techniques are used in this study such features representation, N-gram, Bag of words, traditional and evolutionary features selection, and classification models. This section introduces description for each of them.

1) BINARY FEATURES

Word bigram features [15] are not that commonly used in text classification tasks, the most general term is a bag of words. Probably due to their having mixed and overall limited utility in topical text classification tasks. This likely reflects that certain topic keywords are indicative alone. However, adding bigrams always improved the performance, and often gives better results. The inclusion of word bigram features gives consistent gains on sentiment analysis tasks. For example, using bigrams [16] increases the vector dimension from V to V^2, where V is the vocabulary size. With so many features, care must be taken to include not simply those that are relevant by themselves but only those that are jointly relevant with the rest of the features [17], [18].

2) TERM FREQUENCY (TF) FEATURES

Similar to the BoW method, term frequency [19] is one of the most popular approaches used in textual manipulation. It represents the frequency of a specific term in the correlated part of the text. The frequency is computed by finding the number of keyword occurrences in a document divided by the total number of keywords in the whole document.

3) TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF) FEATURES

Since Term Frequency represents the highest frequent words within a document that have the largest weights, some of these words are insignificant such as the word "the". So, one of the advanced methods to resolve this issue is to use the TF-IDF approach which grants the rare words more weight than the common ones in all documents [20]. TF in TF-IDF approach is the frequency of a particular word in the current document, whereas the IDF assesses how rare the keywords are across all documents.

4) N-GRAMS

N-grams are sequences of adjacent tokens of length *n*. For natural language, n-grams have been used for characters, words, and parts of speech such as nouns, verbs, adjectives, adverbs, conjunctions, and pronouns. This research mainly uses two language models to represent the source code features from a natural language perspective.

5) BAG OF WORDS

Bag of words (BOW) features [21], Bag-of-Words is a textual form approach used in multiple applications like the machine learning classifiers [22]. Each type of text such as a sentence, paragraph, article, or document is treated as a group of words regardless of the syntactical or semantics dependency. It presents the existence of words within a particular text. Bag-of-words features are representations of source code text that describe the occurrence of source code words within a source code file [23].

6) TRADITIONAL FILTER-BASED FEATURES SELECTION

The filter methods [24] are generally used for feature selection. In the filter approach, the features are pre-selected during the pre-processing procedure with regard to a

[1] https://github.com/buptlearner/authorship_attribution

predefined relevance measure which is independent of the performance of the learning algorithm.

The fast Correlation-Based Filter (FCBF) [25] is a multivariate feature selection method where the class relevance and the dependency between each feature pair are considered. Based on information theory, FCBF uses symmetrical uncertainty to calculate dependencies of features and class relevance. FCBF heuristically applies a backward selection process with a sequential search strategy to remove irrelevant and redundant features, starting with the complete feature set. The algorithm stops when there are no features left to eliminate. FCBF was designed for high-dimensional data and has been shown to remove irrelevant and redundant features effectively. However, it fails to consider the interaction between attributes [26].

Fisher's score is one of the most widely used supervised feature selection methods, and the algorithm produces the ranks of the variables in descending order based on Fisher's score. We can choose the appropriate number of features based on the task or problems.

Information gain (InfoGain) [26] calculates the reduction in entropy or disorder from the transformation of a dataset. The InfoGain can be utilized for feature selection by evaluating the information gained from each feature in the target or class variable context. The more information gain value, the more entropy is removed, and the more information the input features (independent features) carry about the target variable or class.

The ReliefF method [26] is a feature selection technique that extends the original Relief algorithm. The original Relief works by randomly sampling a sample from the data and then discovering its nearest neighbor from the same and opposite class. The Relief algorithm is limited to classification problems with two classes or binary class problems. ReliefF adds the ability to deal with multiclass problems and is also more robust and capable of dealing with incomplete and noisy data. In addition, this method may be applied in all situations, has a low bias, includes interaction among features, and may capture local dependencies which other methods miss.

The chi-square test [26] is a statistical test of independence to determine the dependency of two features. It calculates the Chi-square between each feature and the target and selects the desired number of features with the best Chi-square scores. The higher Chi-Squared, the more relevant the feature with respect to class.

7) EVOLUTIONARY FEATURES SELECTION

The genetic algorithm (GA) [27] is a search-based heuristic algorithm primarily used for natural feature selection. The GA has a set of candidate solutions that calls it population. The GA gives the optimal solutions utilizing the survival of the fittest principle. The generated high-fitness populations are represented as chromosomes. The chromosomes provide an acceptable solution to the particular problem. The core process of genetic algorithms is used to mimic genetic evolution, in which the genetics of individuals with the best environmental fit endure through time. The process begins to generate an initial population (or first generation of 70 individuals) of individuals (or chromosomes), each of which has a set of binary genes representing the existence of the features for evaluation. The fitness value of an individual is estimated using a classifier in the form of accuracy or other ML evaluation functions. The fitness function (or objective function) is used to choose the individuals with high accuracy values to reproduce in each iteration.

There are several evolutionary feature selection techniques. This study uses four types of them. The grey wolf optimizer (GWO) [28] is a swarm intelligence optimization algorithm that mimics wolf leadership hierarchy and hunting mechanism. GWO simulates the internal leadership hierarchy of wolves; thus, in the searching process, three solutions can thoroughly assess the position of the best solution. Furthermore, the GWO has a strong exploration ability, preventing the algorithm from falling into the local optimum. For the GWO, the proper compromise between exploration and exploitation ability is very simple to achieve, so it can effectively solve many complicated problems.

Salp Swarm Optimizer (SSA) is an efficient meta-heuristic optimization algorithm that solves various optimization problems in various research areas. It begins by initializing a random position population of salps and computes the fitness value for each; the food source gets the position of the salp that obtained the best fitness value. The best position is always kept as the food position; for that, it never gets lost, although the salps keep exploring and exploiting the search space. As parameter $c1$ declines with iterations, the algorithm explores the search space and exploits it [29].

Harris Hawks Optimizer (HHO) [30] is a swarm intelligence-based meta-heuristic algorithm that simulates the behavior of Harris hawks cooperatively foraging and surrounding prey with multiple strategies. HHO consists of two phases of exploration and exploitation, switching between the two phases through the prey escape energy. As a result, HHO has strong competitive strengths compared with other swarm intelligence optimization algorithms. HHO is easy to use because of its simple structure and no extra parameters except required parameters. Furthermore, the multiple strategies of HHO in the exploitation phase improve the algorithm's local search ability. In addition, HHO simulates the continuous loss process of the prey escaping energy and switches between the exploration phase and exploitation phase based on the value of the prey escape energy, which fits the optimization process and makes the algorithm good performance.

8) CLASSIFICATION TECHNIQUES

With respect to this study, three classifiers, namely Bayesian Network (BN), k Nearest Neighbor (KNN), and Extreme

Learning Machine (ELM) are applied to compare their obtained accuracy. The Bayesian Network (Bayes net) classifier comprises a particular set of features or attributes and a single class variable [31]. The class variable is generally a root in the network, and the features are considered leaves. Unlike the naive Bayesian classifier, the independence assumption [32] is not considered in the Bayes net. The KNN classifier is one of the most commonly used classifiers as it is a simple and effective non-parametric approach for classification [33]. It has one parameter (K) to identify the number of selected nearest neighbors [34]. However, searching for the value of (K) is difficult, especially with high-dimensional data. Generally, the K parameter in the KNN classifier is selected empirically. Depending on each problem domain, various numbers of nearest neighbors are tested. Ultimately, the parameter with the best accuracy is chosen to define the classifier as stated in the study [35]. Further, ELM classifier [36] has three parameters of hidden nodes, i.e., input weights, biases, and output weights. Both input weights and biases are randomly assigned, while the simple generalized inverse operation can state the output weights analytically. The number of hidden nodes required to be defined. Unlike other traditional classifiers, ELM provides faster learning speed and better performance with minimum human intervention.

B. RELATED WORK

Most of the research works in authorship attribution of source code focused on C/C++, few studies focused on Java source code [37] and Java repositories are much fewer compared to C/C++. Many studies [6], [11], [12], [13], [14] proposed different approaches to accurately identify authors for Java source code.

Ding and Samadzadeh [11] used Krsul's C metrics for Java, which included layout, style, and structure, and utilized byte-level n-gram profiles to measure their approach. The results show that 48 metrics were detected as contributing metrics out of 56 metrics. But, their study lacks to provide a ranking for all final features. Also, the study conducted by Lange and Mancoridis [12] found that the study [11] used only scalar quantity metrics extracted from source codes, their study formulated 18 metrics as histogram distributions. It used different code metrics and genetic algorithms to identify authors in software forensics. Similarly, the study [13] only selected four metrics (leading spaces, leading tabs, line length, words per line) out of the 18 metrics in the study [12], they applied the genetic algorithm to discretize metrics. But this small feature set is also non-reproducible and did not give sufficient details to have the final features set. The experiments were done on 20 authors and over 194 750,000 lines of Java source codes. Moreover, the study [6] summarized the previous classification techniques for the above studies by using machine learning and information retrieval. The study provided 90% and 85% accuracy. Other studies focused on identifying authors for JavaScript code instead of Java such as the study [14], which proposed an approach to extract some structural features from AST and achieved 85% accuracy for 34 authors.

Several research works focused on exploring authorship attribution in Java. The studies [38], [39] employed different ML techniques to identify authors based on using different sets of features. Both of these studies demonstrated their approaches on the Java GitHub dataset. The study [38] used a back propagation neural network based on particle swarm optimization, it used 19 features. Based on their approach, the reported experimental achieved an accuracy of 76% by using stochastic gradient descent (SGD) as a classifier and Abstract Syntax Tree (AST) as features selection.

Furthermore, in the research conducted by [39], a deep neural network technique called ICodeNet was utilized for performing tasks related to source code files at the file level. ICodeNet combines an encoder based on the ImageNet-trained VGG (Visual Geometry Group) model with a shallow neural network.

Some studies [40], [41], [42] proposed using different NLP classification of text documents, the study [40] used NLP approach based on Naive Bayes and N-Gram features. The study [41] used PrimePatNet87 as prime pattern and tunable q-factor wavelet transform techniques for automated accurate EEG emotion recognition. Also, the study [42] used Meniscal tear and ACL injury as detection model based on AlexNet and iterative ReliefF.

The previous investigations utilized a minimum set of Java code features (i.e. 48 or 18), primarily focusing on some of syntactic and stylistic characteristics which cover only certain aspects of code. On other hand, this research aims to deeply explore a comprehensive range of features that reflect and cover all the parts of the source code as identifiers, variables, data types, class and method names, control structures, data structures, reserved word, comments, and etc. It mainly treats the source code lexicon as regular natural language text. To improve the accuracy of authorship attribution models, this study capitalizes on the benefits of using a Genetic Algorithm (GA) for feature selection and K-Nearest Neighbors (KNN) as a classifier. It introduces two language models (uni-gram and bi-gram) along with three feature representations, namely binary, Term Frequency (TF), and Term Frequency-Inverse Document Frequency (TF-IDF).

III. MATERIALS AND METHODS

This section provides an overview of the used dataset, the process of extracting and representing features, the proposed methodology, and the metrics used for evaluation.

A. DATASET DESCRIPTION

GitHub 2016-Authorship Attribution: The dataset was introduced by [38], the authorship dataset comprises 3,022 Java files with 40 authors. The minimum file number that an author contributes is 11, and the maximum is 712. Also, it has 65,200 Unique tokens and 220,000 unique paths. Also, the average line length of these Java files is 98.63, ranging from 16 lines to 11,418 lines. These extracted features form

a feature line, representing the Java source file belonging to its corresponding author. and it consists of 40 authors who wrote 3,022 files. Figure. 1 illustrates the distribution of the instances among 40 authors (class labels).

B. PROPOSED APPROACH

The proposed approach uses the GA-KNN framework to reduce the feature space dimension and select the optimal subset of features, which enhances the performance of the classification process. It involves utilizing an evolutionary model to select relevant features, then employing the optimal feature set in conjunction with K-nearest neighbors (KNN) for classification. The primary focus of this study is on authorship attribution, where an evolutionary-based classification technique for authorship attribution has been proposed. Initially, the feature set was extracted from the authorship dataset, preceded by appropriate data pre-processing. In this step, 1500 features set (tokens) were extracted.

The dataset needs to be given in a way that is appropriate for the ML phase. Therefore, extracting and vectorizing the most essential text features is necessary. The two-dimensional matrix used to describe the vector space has rows for the Java files and columns for the features. The matrix entries represent the weights of the features in their respective authors.

Subsequently, feature vectors were generated by applying binary, term frequency (TF), and term frequency-inverse document frequency (TF-IDF) feature extraction techniques to all Java files. Furthermore, the process included completing feature selection using uni-gram and bi-gram language models. Based on our experiments, the number of extracted features in uni-gram and bi-gram models were 39061 and 223664, respectively. Once the relevant features were identified, traditional machine learning classifiers were investigated to identify authors. Additionally, evolutionary-based classification methods were explored in this research to improve the performance of authorship attribution. The structure of the research approach used in this study is illustrated in Figure 2.

1) DATA PREPROCESSING

This paper employs a two-step approach for the pre-processing task, which holds significant importance in extracting features.

1) Merging all the Java files for all authors in one document.
2) Eliminating unnecessary characters [43], This step includes extra spaces, punctuation marks, etc. which used in the dataset are meaningless and unnecessary for our data analysis [44]. Such punctuation does not help to identify the writing style of the authors.

2) FEATURES EXTRACTION

This step includes uni-gram and bi-gram models, both models are described as in the following subsections. We focused on un-igram and bi-gram rather than tri-gram, because tri-grams are so sparsely distributed, and in turn it rises the problem of dimensionality comparing with bi-gram. Moreover, most of the research as in the study [45] recommended on the combination of unigrams and bigrams using different feature selection techniques.

a: UNI-GRAM LANGUAGE MODEL

The input to the proposed detection model is a vector of unigrams after binary, TF, and TFIDF has been applied, representing a single document from a user. The token output is the raw output given by the above method as a vector of tokens while the n-grams output will generate unigrams, bigrams, TF, and TF-IDF frequencies of the traversal The lexical features are generated from the raw source code including features such as word unigram term frequency, normalized keyword frequency for each of the common keywords, these features are used to show a programmer's preference for a specific type of keywords. Uni-gram [46], [47], [48] is using word uni-gram feature sets as a single-word token from the source code.

b: BI-GRAM LANGUAGE MODEL

A bigram language model [49] takes advantage of the correlation between the two neighboring words while a unigram treats each word as independently generated. Experiments in authorship attribution have shown that bigram language models can achieve dramatic improvements over simple unigram models.

Known and unknown texts are represented as vectors of term weights. We may understand a vector as a set of attributes and values, regardless of their order or sequence. Vectors of known texts of the same author are summed together into the same class. The elements of these vectors, the terms, are bigrams associated with their frequency of occurrence, being a bigram [50] in this context a sequence of two words as they appear in the text. A word is defined as a sequence of characters between spaces and the weight is just its frequency of occurrence after the corpus has been normalized. Bigram model features are greater than the unigram model for the reason of that the number of tokens in bi-gram is greater than in uni-gram since it has two adjacent words rather than having single words.

This paper considers each word of source code as a single feature. For instance, the data types, names for the methods, classes, identifiers, and reserved words are all considered features. It takes a bag of words from a code file as input, transforms them into a single numerical vector, and predicts a probability for each known developer to be the author of the Java code file. The BoW is used to select the most 1500 frequent words among all extracted features in both uni-gram and bi-gram models. The GA selects features in a range from 500 to 1500. The selection criteria are based on this range and based on the experimental setting.

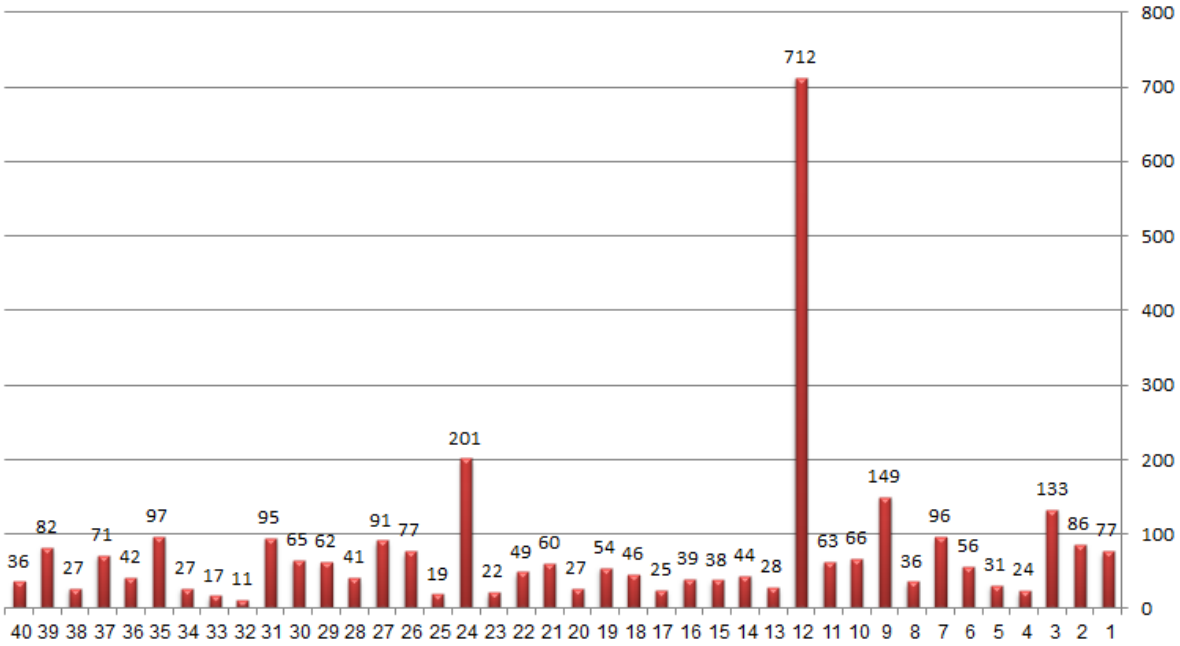


FIGURE 1. Data distribution among 40 class labels.

3) FEATURES REPRESENTATION

This paper uses three techniques for feature representation (binary, TF, and TF-IDF). To preserve inter-token relations [51]. This paper generates binary, TF, and TF-IDF vectors using unigrams and bigrams to represent files for the Java source code. However, due to the combinatorial nature of n-grams, the number of terms grows quickly with the number of documents as well as the number of tokens that form an n-gram.

4) FEATURES SELECTION TECHNIQUES

This paper uses four evolutionary feature selection methods genetic algorithm, grey wolf optimizer, Salp Swarm Optimizer, and Harris Hawks Optimizer as they described in the subsection II-A.

5) CLASSIFICATION TECHNIQUES

With respect to this study, three classifiers have used, namely Bayesian Network (BN), K Nearest Neighbor (KNN), and Extreme Learning Machine (ELM). Based on the comparison, KNN has the highest obtained accuracy. KNN is a commonly utilized classifier. This choice was made due to its simplicity and efficacy as a non-parametric classification approach [33]. It has one parameter (K) to identify the number of selected nearest neighbors [34] to predict the class labels of the unknown samples. The value of this parameter has a significant impact on classification performance. However, searching for the value of (K) is difficult, especially with high-dimensional data. Generally, the K parameter in the KNN classifier is selected empirically. Depending on each problem domain, various numbers of nearest neighbors are tested, and the parameter with the best accuracy is chosen to define the classifier. The description for those classifiers is given in the subsection II-A.

C. EVALUATION MEASURES

To measure the performance of the proposed model, several evaluation metrics were used: accuracy, sensitivity, specificity [52], F-measure, and geometric mean (G-mean) [35], [53]. The evaluation measures that used in this paper are shown in the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

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

$$Sensitivity/recall = TP/(TP + FN) \tag{3}$$

$$Specificity = TN/(TN + FP) \tag{4}$$

$$precision = TP/(TP + FP) \tag{5}$$

$$G - mean = \sqrt{(precision * recall)} \tag{6}$$

The details descriptions of the used metrics can be found in the studies [35], [52], [53].

IV. RESULTS AND DISCUSSION

This section clarifies the programming language and tools used in this research. Also, the parameter settings for the evolutionary techniques and classification techniques are discussed. Moreover, it presents and analyzes the experimental outcomes of different evolutionary models using uni-gram and bi-gram approaches on multiple dataset versions, including binary, TF, and TF-IDF, to demonstrate the algorithm's evaluated performance. Furthermore, the

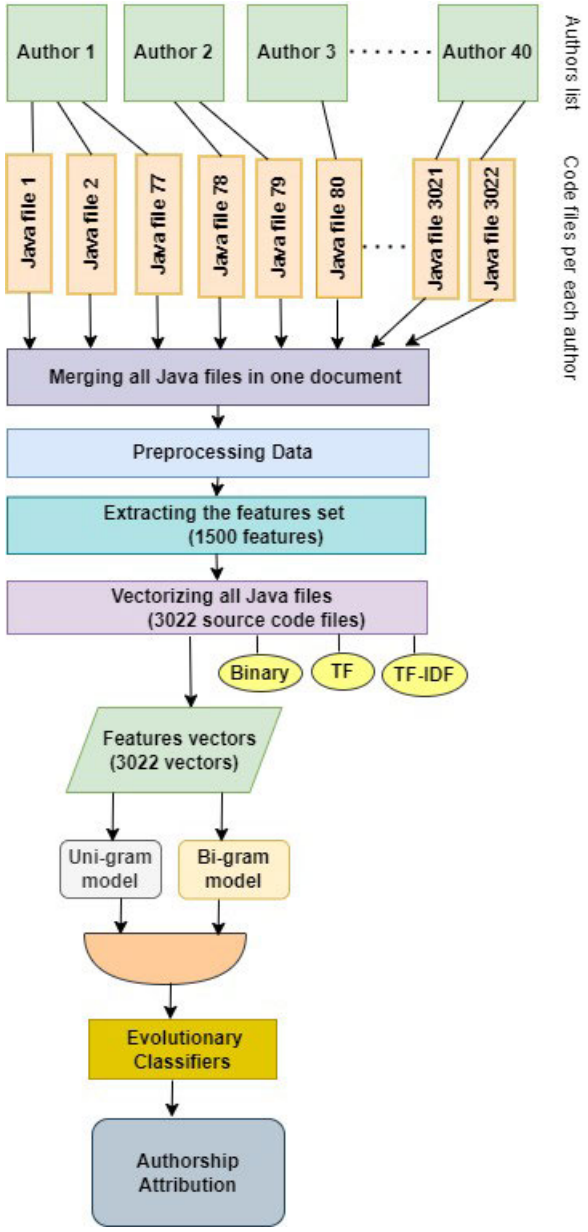


FIGURE 2. The proposed methodology by selecting the most 1500 features.

study compares the proposed model with a filter-based feature selection method, traditional classification models, and the performance of other related work.

A. PROGRAMMING LANGUAGE AND TOOLS

The suggested method is implemented using the Matlab R2019a tool on an Intel(R) Xeon(R) computer running at 2.20 GHz with 64 GB of RAM. The ease of use and accessibility to supporting toolboxes, including the parallel toolbox, which accelerates computing, are two benefits of using Matlab. Additionally, Matlab manages complicated data for difficult engineering challenges and simulations. The proposed and the compared approaches are implemented using the same platform and programming language to get fair comparisons.

B. SENSITIVITY ANALYSIS AND PARAMETER SETTINGS

The parameters used in the experiments were carefully established based on the sensitivity analysis. The analysis set up the population size with different values; these are 10, 20, 30, 50, and 70 search agents. The number of iterations is set to 100 based on the investigation of the studies [54], [55], [56], [57]. Figure. 4 shows the results of the measurements made using various datasets and the size mentioned above of population parameters. Each dataset used in this study is randomly split into a training set (70%) and a testing set (30%) [58]. Since the datasets are multi-class labels with different distribution ratios, we used a stratified sampling technique to ensure that the same proportion of each class label is included in both sets. All algorithms were performed ten times with a random seed, and the average was used.

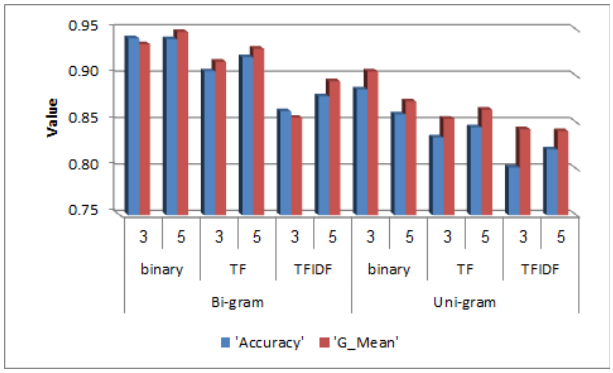
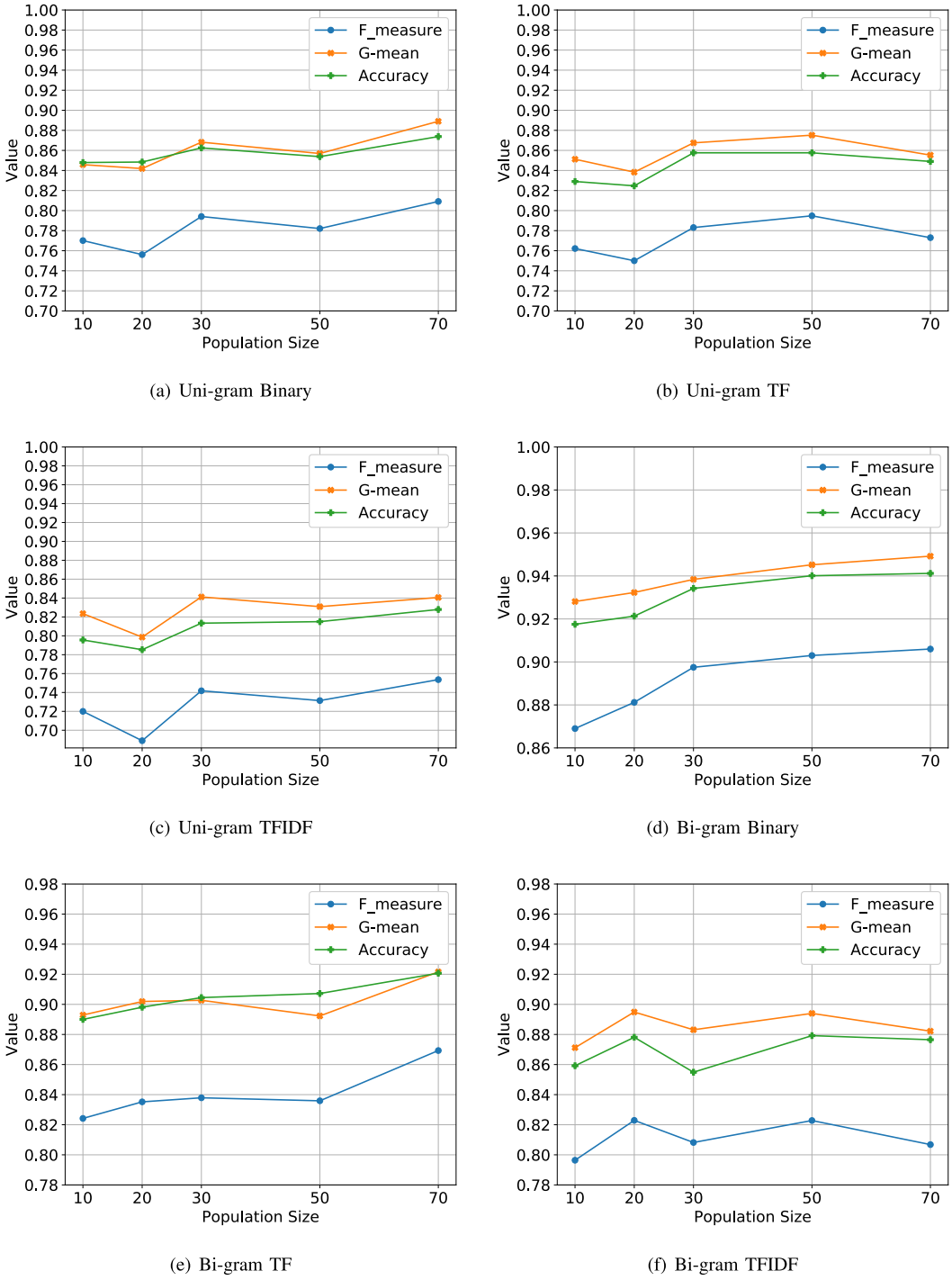


FIGURE 3. Comparison of the classification accuracy, Gmean for different values for K in the KNN classifier.

The most popular classifier using the various datasets in the UCI repository is KNN. In contrast, the SVM classifier is frequently employed in intrusion detection systems and datasets from the medical profession, including those for diagnosing artery disease and cancer. Therefore, [59] looked into using the SVM method to show the salient features under time limitations on medium-sized feature datasets. The KNN classifier, however, is the most used classifier and has advantages when employed with high-dimensional datasets [60]. Various random K values were employed, and $k = 5$ was eventually selected due to its superior classification performance when it compared to the other values tested as in [61] and [62]. Notably, it is worth mentioning that a majority of prior studies in the literature have also advocated using the same K value for the K-nearest neighbors (KNN) classifier. For that, we adopt the KNN classifier as the base classifier in this work with ($k = 5$) as indicated in the studies [60], [63]. In addition, Figure. 3 demonstrates the sensitivity of using 3 and 5 for the value of K, which proves the suitability of using the KNN with $k = 5$.

It is abundantly evident that the population size of 70 produces the highest conduction measure values. For the remaining experiments, we use this size as a result. The



(a) Uni-gram Binary

(b) Uni-gram TF

(c) Uni-gram TFIDF

(d) Bi-gram Binary

(e) Bi-gram TF

(f) Bi-gram TFIDF

FIGURE 4. **Sensitivity results over different population size.**

parameter settings that will be used in this study are shown in Table.1. The parameters of the comparison algorithms are shown in Table.2

C. THE EVOLUTIONARY MODEL'S RESULTS

In this section, we employed several evolutionary algorithms for selecting the best informative feature sets from the original 1500 features. We adopt these algorithms; Genetic Algorithm (GA), Grey Wolf Optimizer (GWO), Salp Swarm Algorithm (SSA), and Harris Hawk's Optimizer (HHO) in the wrapper-based approach. The results of the evolutionary models based on the uni-gram extracted datasets will be introduced and analyzed in subsection IV-C1. The bi-gram results are presented and discussed in subsection IV-C2.

TABLE 1. List of the used parameters in the experiments.

No.	Parameter	Value
1.	Classifier	KNN (K=5)
2.	Training Data	70%
3.	Testing Data	30%
4.	Population Size	70
5.	Max Number of iterations	100

TABLE 2. The parameter settings of the compared algorithms.

Algorithm	Parameter	Value
GA	Crossover percentage	0.8
	Mutation percentage	0.3
	Mutation rate	0.02
	Selection scheme	Random
	Tournament size	3
	Beta	8
SSA	C_2	random value $\in$ [0 1]
	C_3	random value $\in$ [0 1]
	Upper bound	1
	Lower bound	0
GWO	Convergence constant α	[2 0]
HHO	Upper bound	1
	Lower bound	0
	Transfer function	S2

The evolutionary-based feature selection will be applied to all versions of the dataset; binary, TF, and TF-IDF versions for both the uni-gram and bi-gram models. This paper uses Gmean as an evaluation measure for particular imbalanced data sets, the highest mean results are more robust in the imbalance dataset more robust classifier. In this section, we present the evolutionary-based feature selection to all dataset versions, including binary, TF, and TF-IDF representations, for both uni-gram and bi-gram models.

1) THE UNI-GRAM RESULTS

Table 3 and Table 4 show obtained classification results and best fitness values for the uni-gram language model. Among all the four used evolutionary feature selection techniques (GA, GWO, SSA, HHO), GA shows superior accuracy performance in each type of feature. The highest achieved accuracy of GA was reported as 86.02%, 84.75%, and 83.69% for Binary, TF, and TF-IDF respectively. The number of selected features shows a superior reduction rate in the dataset dimension, which minimizes the dimension for the binary version by 50%, 49% for the TF version, and 48% for the TFIDF version. Additionally, the GA achieves minimum fitness values during iterations, which proves the GA's ability to avoid falling into local optima. In contrast, the GA saves resources and computations by taking minimum running time compared with other evolutionary algorithms.

As clearly shown in Fig. 5, the GWO, SSA, and HHO were stuck in the local optima region based on their exploration and exploitation capabilities. In contrast, the GA avoided falling into local optima by using crossover and mutation techniques, making the generated solutions more diverse and covering more positions in the search space. On the other hand, the nature of the dataset plays an essential role in the classification process. In the binary model, the data will be represented in 0's, and 1's format, which facilitates the classification process [64].

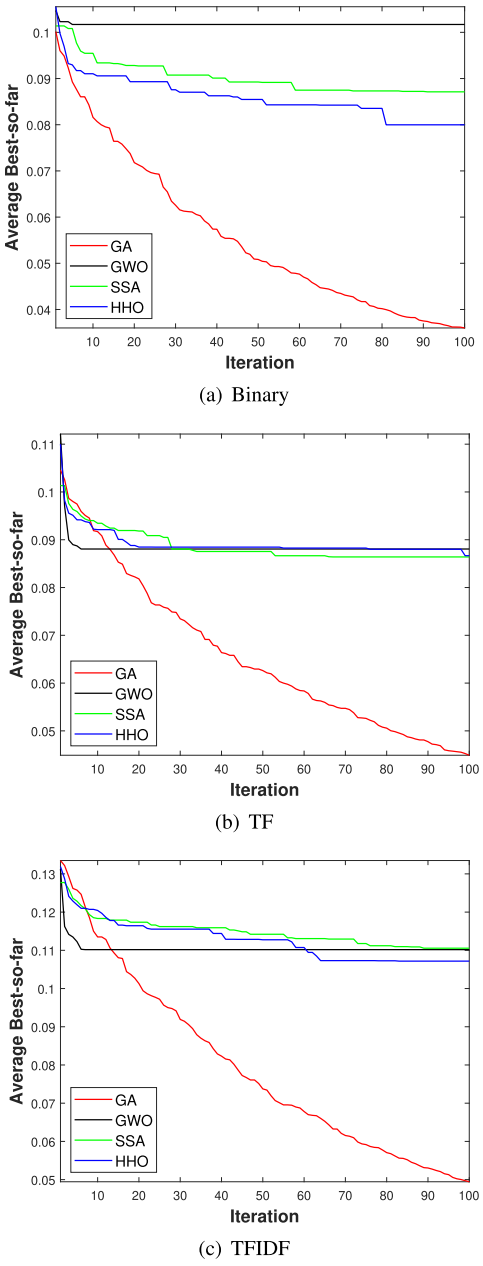


FIGURE 5. Convergence curves for the evolutionary models on the UniGram models.

2) THE BI-GRAM RESULTS

Table 5 and Table 6 show the classification results and the best fitness values for Bi-gram language model. Like in the Uni-gram model, GA shows superior accuracy performance in each type of feature. The highest achieved accuracy of GA was obtained as 93.72%, 91.37%, and 88.05%, with 687, 723, and 724 selected features for Binary, TF, and TF-IDF respectively. Similarly, the outcomes show that the performance of GA with Binary features is better than TF and TF-IDF. Obviously, the GA has the highest accuracy due to several reasons. First, GA is efficiently used to determine the minimal subset of features for which the different classes are best distinguished during classification [24]. Second,

TABLE 3. Evolutionary results using UniGram language model.

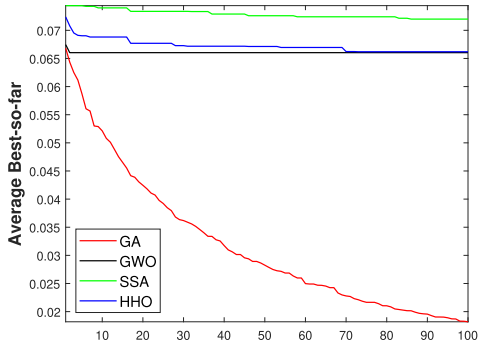
Benchmark	Algorithms	Measure	Sensitivity	Specificity	F_measure	Accuracy	G-mean
Binary	GA	AVG	**0.7639**	**0.9964**	**0.7775**	**0.8602**	**0.8522**
		STD	0.0128	0.0002	0.0131	0.0069	0.0119
	GWO	AVG	0.7287	0.9955	0.7436	0.8263	0.8323
		STD	0.0177	0.0001	0.0148	0.0054	0.0131
	SSA	AVG	0.7578	0.9960	0.7731	0.8438	0.8499
		STD	0.0080	0.0001	0.0060	0.0052	0.0035
	HHO	AVG	0.7599	0.9956	0.7648	0.8302	0.8483
		STD	0.0126	0.0002	0.0134	0.0081	0.0133
TF	GA	AVG	**0.7766**	**0.9961**	**0.7840**	**0.8475**	**0.8663**
		STD	0.0052	0.0001	0.0044	0.0044	0.0035
	GWO	AVG	0.7655	0.9959	0.7774	0.8412	0.8567
		STD	0.0089	0.0002	0.0101	0.0060	0.0082
	SSA	AVG	0.7303	0.9954	0.7382	0.8216	0.8336
		STD	0.0109	0.0003	0.0133	0.0122	0.0072
	HHO	AVG	0.7604	0.9958	0.7746	0.8399	0.8538
		STD	0.0036	0.0002	0.0065	0.0074	0.0034
TFIDF	GA	AVG	**0.7744**	**0.9957**	**0.7770**	**0.8369**	**0.8638**
		STD	0.0069	0.0001	0.0065	0.0032	0.0071
	GWO	AVG	0.7042	0.9944	0.7234	0.7886	0.8171
		STD	0.0076	0.0003	0.0133	0.0095	0.0101
	SSA	AVG	0.7058	0.9946	0.7156	0.7920	0.8158
		STD	0.0118	0.0003	0.0092	0.0116	0.0095
	HHO	AVG	0.6858	0.9944	0.6959	0.7864	0.7971
		STD	0.0122	0.0003	0.0121	0.0092	0.0098

TABLE 4. Best fitness values, number of selected features, time using UniGram language model.

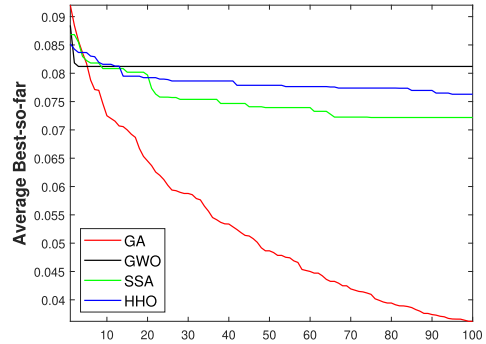
Benchmark	Algorithms	Measure	Best fitness	#Features	Time
Binary	GA	AVG	**0.0360**	**751.0000**	**42524.5660**
		STD	0.0016	20.9550	856.9171
	GWO	AVG	0.1017	966.0000	64201.0028
		STD	0.0040	165.6516	2830.2937
	SSA	AVG	0.0871	961.6000	63803.5853
		STD	0.0052	14.9161	59.1863
	HHO	AVG	0.0800	1024.4000	85606.9106
		STD	0.0046	59.1631	653.4792
TF	GA	AVG	**0.0449**	**766.4000**	**43415.2520**
		STD	0.0026	13.4429	499.4275
	GWO	AVG	0.0881	1329.0000	72438.2369
		STD	0.0050	39.6989	1224.6238
	SSA	AVG	0.0864	966.8000	63839.8975
		STD	0.0013	17.7313	92.8723
	HHO	AVG	0.0867	1034.4000	88427.8905
		STD	0.0030	53.4025	780.6461
TFIDF	GA	AVG	**0.0494**	**782.0000**	**44585.4301**
		STD	0.0039	21.6128	736.5181
	GWO	AVG	0.1102	1315.2000	72049.2324
		STD	0.0089	109.7196	3084.8998
	SSA	AVG	0.1106	974.6000	64509.4731
		STD	0.0015	17.3025	129.2863
	HHO	AVG	0.1072	1006.0000	86517.5081
		STD	0.0037	91.1165	1410.4692

GA is repetitively performed over multiple generations, and reproduction is performed to get individuals that are the best fit for a certain domain, the only individuals with high fitness will maintain over time. Based on their capacities

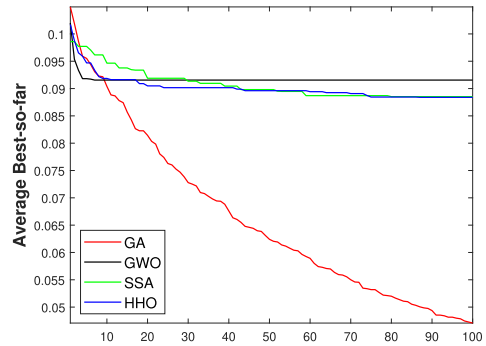
for exploration and exploitation, the GWO, SSA, and HHO were firmly entrenched in the local optima zone, as is seen in Fig. 1. In contrast, the GA [65] used crossover and mutation approaches to increase the diversity of the generated solutions and widen the coverage of the search space, preventing it from settling into a local optimum. On the other hand, the classification process [66] is greatly influenced by the dataset's nature. The binary model simplifies the classification process by representing the data in a format of 0s and 1s.



(a) Binary



(b) TF



(c) TFIDF

FIGURE 6. Convergence curves for the evolutionary models on the BiGram models.

Finally, the Friedman test method was used to statistically compare each independent run and highlight the significance of the results that were not produced by chance. The well-known non-parametric statistical test method known as Friedman's test is always employed to assess the performance levels of algorithms. Also, in order to determine whether there is a significant difference between the results of the various algorithms, Friedman's test is used [67]. Table. 7 illustrates the F-test's ranking for each of the used algorithms in the adopted data sets. It is clearly shown the GA takes the first rank. GWO was in second place, while the SSA and HHO ranked third.

This rank proves that the GA significantly improves the authorship classification problem comparing with others evolutionary features selection approaches.

D. COMPARISONS WITH FILTER-BASED FEATURE SELECTION METHODS

In this subsection, a comparison between the used wrapper-based FS approach, which is GA, is compared with the traditional filter-based FS methods. Namely Fast Correlation Based Filter (fcbf) [68], fisher score (fisher) [69], information gain (infogain) [70], relief f-based (relief) [71], and chi-square (chi2) [72]. The conducted accuracy results are illustrated in Table 8. The top 50 ranked features were used in this experiment based on analysis of the number of used features, while the number 50 shows the best results over these filter methods as illustrated in Figure. 7. As clearly shown, the GA outperforms all other filter-based methods regarding classification accuracy for both Uni-gram and Bi-gram models. These results prove the efficiency of the wrapper-based methods in solving the FS problem. The F-test ranking results are shown in Table 9. It also shows the GA placed in the first rank compared with other methods.

E. COMPARISONS WITH TRADITIONAL CLASSIFIERS

Multiple types of traditional classifiers are used in source code authorship attribution. With respect to this study, three classifiers, namely Bayesian Network (BN), k Nearest Neighbor (KNN), and Extreme Learning Machine (ELM) are applied to compare their obtained accuracy.

To show the strength of the GA with KNN against multi-class classifiers, Table. 10 shows the conducted results of comparing the GA-KNN with ELM, KNN, and BN. These classifiers were used as a multi-class classifier on the used dataset. The results show the GA's superior performance in classifying the source code authorship in the Bi-gram model. While the Bayse Net performs better in the Uni-gram model. Here, we should notice that the traditional classifiers use all feature sets in the dataset, while the GA selects the best informative set. Table 11 shows the F-test ranks for the traditional classifiers compared with the GA-KNN. Instead of the good performance of the Bayse Net on the Uni-gram model, the GA-KNN still has the first position in the ranking.

F. COMPARISONS WITH RELATED WORKS

To validate the proposed model, this study compares the performance of our results with other related approaches [38],

TABLE 5. Evolutionary results using BiGram language model.

Benchmark	Algorithms	Measure	Sensitivity	Specificity	F_measure	Accuracy	G-mean
Binary	GA	AVG	**0.8945**	**0.9984**	**0.8959**	**0.9372**	**0.9410**
		STD	0.0075	0.0001	0.0042	0.0048	0.0048
	GWO	AVG	0.8419	0.9975	0.8503	0.9038	0.9077
		STD	0.0055	0.0001	0.0081	0.0049	0.0078
	SSA	AVG	0.8452	0.9975	0.8487	0.9025	0.9083
		STD	0.0176	0.0003	0.0191	0.0103	0.0107
	HHO	AVG	0.8610	0.9979	0.8714	0.9184	0.9210
		STD	0.0109	0.0001	0.0140	0.0060	0.0075
TF	GA	AVG	**0.8572**	**0.9978**	**0.8536**	**0.9137**	**0.9119**
		STD	0.0120	0.0001	0.0120	0.0032	0.0108
	GWO	AVG	0.7563	0.9964	0.7669	0.8608	0.8472
		STD	0.0116	0.0002	0.0107	0.0090	0.0089
	SSA	AVG	0.7915	0.9967	0.7961	0.8706	0.8755
		STD	0.0081	0.0001	0.0065	0.0052	0.0071
	HHO	AVG	0.8034	0.9968	0.8065	0.8744	0.8830
		STD	0.0090	0.0001	0.0050	0.0049	0.0068
TFIDF	GA	AVG	**0.8133**	**0.9969**	**0.8233**	**0.8805**	**0.8917**
		STD	0.0223	0.0003	0.0217	0.0125	0.0140
	GWO	AVG	0.7516	0.9957	0.7720	0.8365	0.8547
		STD	0.0108	0.0002	0.0128	0.0072	0.0064
	SSA	AVG	0.7647	0.9959	0.7760	0.8410	0.8576
		STD	0.0061	0.0002	0.0051	0.0069	0.0060
	HHO	AVG	0.7532	0.9959	0.7652	0.8408	0.8543
		STD	0.0088	0.0001	0.0088	0.0035	0.0073

TABLE 6. Best fitness values, number of selected features, time using BiGram language model.

Benchmark	Algorithms	Measure	Best fitness	#Features	Time
Binary	GA	AVG	**0.0182**	**687.8000**	**41105.0258**
		STD	0.0012	26.2755	707.7275
	GWO	AVG	0.0660	817.4000	60611.3807
		STD	0.0029	160.7228	3477.6112
	SSA	AVG	0.0720	946.4000	62823.0141
		STD	0.0016	6.4153	148.4804
	HHO	AVG	0.0662	799.4000	81754.3885
		STD	0.0016	81.0339	879.0306
TF	GA	AVG	**0.0362**	**723.6000**	**42171.7353**
		STD	0.0047	19.6706	500.5453
	GWO	AVG	0.0812	1023.6000	67534.0110
		STD	0.0025	144.7213	1643.2320
	SSA	AVG	0.0722	961.4000	63233.8962
		STD	0.0028	15.5578	46.2743
	HHO	AVG	0.0763	982.2000	86521.8672
		STD	0.0010	51.7060	907.9052
TFIDF	GA	AVG	**0.0471**	**724.8000**	**42217.8461**
		STD	0.0054	18.4920	481.9084
	GWO	AVG	0.0916	1229.0000	70307.0740
		STD	0.0056	99.9733	2935.9003
	SSA	AVG	0.0885	958.4000	63357.8416
		STD	0.0021	17.3410	61.9601
	HHO	AVG	0.0884	939.6000	86616.3136
		STD	0.0010	76.2936	907.8417

[39] as in Table 12. The best-achieved accuracy was reported at 85.4% and 76% for the studies [38], [39] respectively. Whereas, the highest accuracy obtained by the proposed approach has resulted as 86.2% and 93.7%

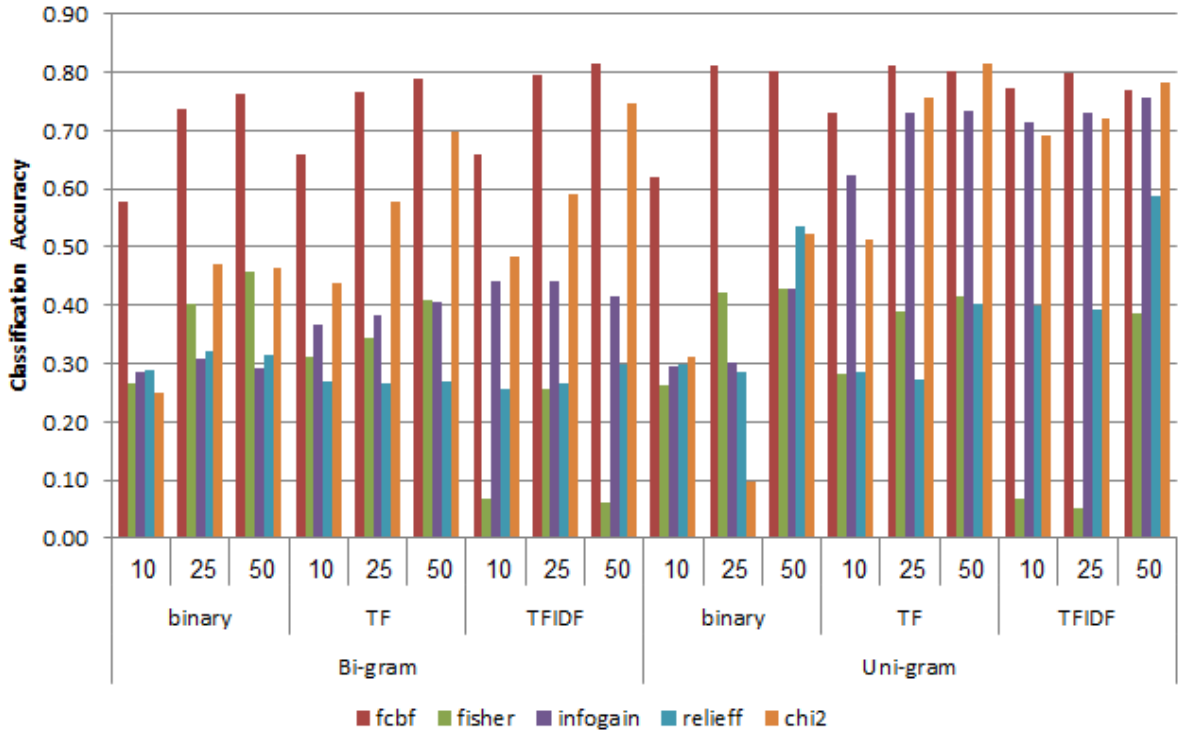


FIGURE 7. Comparison of the classification accuracy for different numbers of features in the filter-based methods.

TABLE 7. A summary of the ranking results obtained by applying Friedman's test to the evolutionary results.

Algorithm	Ranking
GA	1.0000
GWO	3.3333
SSA	2.8333
HHO	2.8333

TABLE 8. A comparison results in terms of classification accuracy using the evolutionary model and filter-based FS approaches.

Model	Benchmark	GA	fcbf	fisher	infogain	relieff	chi2
Bi-gram	binary	**0.9372**	0.7641	0.4564	0.2911	0.3164	0.4642
	TF	**0.9137**	0.7894	0.4090	0.4057	0.2679	0.6968
	TFIDF	**0.8805**	0.8159	0.0628	0.4157	0.2977	0.7453
Uni-gram	binary	**0.8602**	0.8015	0.4278	0.4289	0.5358	0.5237
	TF	**0.8475**	0.8004	0.4168	0.7332	0.4024	0.8148
	TFIDF	**0.8369**	0.7696	0.3870	0.7552	0.5877	0.7839

TABLE 9. A summary of the ranking results obtained by applying Friedman's test to the filter-based results.

FS method	Ranking
GA	1.0000
fcbf	2.3333
fisher	5.1667
infogain	4.6667
relieff	5.0000
chi2	2.8333

for the uni-gram and bi-gram models correspondingly. Remarkably, the results reflect the significant improvement of prediction accuracy for source code authorship attribution

TABLE 10. Comparison between the GA-KNN with traditional classifiers in terms of classification accuracy.

Model	Benchmark	KNN	ELM	Bayse Net	GA-KNN
Bi-Gram	Binary	0.8857	0.8371	0.8820	**0.937218**
	TF	0.8587	0.7594	0.8720	**0.913698**
	TFIDF	0.8306	0.8091	0.8670	**0.880474**
Uni-Gram	Binary	0.8123	0.7497	**0.8710**	0.860194
	TF	0.8145	0.6030	**0.8680**	0.847466
	TFIDF	0.7735	0.7228	**0.8640**	0.836892

TABLE 11. A summary of the ranking results obtained by applying Friedman's test to the results of the traditional classifier.

Method	Ranking
GA	1.5000
Bayse Net	1.6667
KNN	2.8333
ELM	4.0000

TABLE 12. Comparison between the related approaches on Java 40 authors dataset.

Method	Classifier	Optimizer	Features representation	Accuracy
Visual Geometry Group	Voting	RMSProp	ImageNet	73.9%
Visual Geometry Group	CNN	RMSProp	ImageNet	82.8%
Visual Geometry Group	LSTM	RMSProp	ImageNet	85.4%
BPNN-SGD	BPNN	SGD	AST	76%
Uni-gram Langauge model	KNN	GA	Binary	**86.2%**
Bi-gram Language model	KNN	GA	Binary	**93.7%**

with reduced fewer source code features. Notably, the proposed language model outperforms the previous work for identifying Java programmers within a reasonable spent time.

V. SUMMARY

Source code authorship attribution is a challenging task increasingly being investigated in software engineering research because of its importance in different domains like software forensics, software laws, software education, etc. Particularly, it has significant implications in various aspects like software plagiarism, malicious code, software attack, etc. Using features selection significantly impacts classification accuracy. Therefore, using NLP techniques is very important to extract different source code features, which would be extremely helpful in improving the accuracy of identifying authors.

This study aims to explore the impacts of feature selection techniques on improving the identification of source code authorship attribution. Evolutionary algorithms show a superior performance in solving feature selection optimization problems. For that, this paper investigates and evaluates well-known optimizers (GA, GWO, SSA, and HHO) in order to find the proper well-performed optimizer for the source code authorship attribution. Three feature representations (Binary, TF, and TF-IDF) are implemented and assessed for this purpose, along with the unigram and bigram language models. After that, the most effective optimizer used in this experiment. We also assess the adopted optimizer with well-known filter-based feature selection techniques and with the traditional classifiers using the entire feature set, and we compare the results to those reported in the literature to demonstrate the effectiveness of the evolutionary algorithm in the wrapper-based approach.

In summary, GA-KNN is adopted in this study since it performs better than other evolutionary optimization algorithms. Using the GA-KNN as an evolutionary classifier significantly surpasses traditional classifiers (KNN, Bays Net, and ELM) and reduces features to about 700 instead of using the whole set of features (i.e., 1500 features). In contrast, the GA-KNN surpasses the work of the kinds of literature. Based on the obtained results, the study proves that using evolutionary feature selection techniques significantly improves the accuracy of authorship attribution of source code in both uni-gram and bi-gram language models.

On the other hand, the study shows that some traditional classifiers outperform the accuracy of the different utilized standard feature selection. But, generally, the study concludes that using the overall features set will consume more time and resources, which is not preferred in ML models.

This research work treats the source code as normal natural language text. We consider the comment lines along with the source code lines. The proposed model could open the doors and channels to extensively explore the source code features from the natural language side rather than only depending on few sets of the syntactic and semantic aspects. This work uses the bag of words to construct the features for both source code and comments views. This research holds significance in assessing how different classifiers perform when considering both traditional and evolutionary features. The adoption of this proposed model stands to make a substantial contribution to the field of software engineering, ultimately improving the accuracy of source code authorship attribution.

VI. LIMITATIONS

We acknowledge that our study has some limitations. First, as our method allows for both training and attribution at different types of feature representation, our ground truth was limited to binary, TF, and TF-IDF. Second, the used dataset contains only 40 authors in total, we plan in future work to further investigate other larger data sets.

VII. CONCLUSION AND FUTURE WORK

The identification of the source code's author is a crucial task in various fields of software engineering, including computer security, computer laws, and education. This research paper proposes an evolutionary model for selecting features that aim to enhance the accuracy of authorship attribution. The model introduces two language models to identify the 1500 most frequent features from a pool of extracted features. The approach utilizes the K-nearest neighbors (KNN) classifier and Genetic Algorithm (GA) as an evolutionary feature selection method. The experimental results demonstrate a significant improvement in accurately identifying the programmer responsible for each Java source code compared to other relevant studies.

One potential avenue for future research involves broadening the scope of this study to include more datasets related to the attribution of source code authorship, such as Google Code Jam (GCJ). Also, one of important directions in future is to use other N-gram approaches such as tri-gram model. Further, an essential expansion would be creating an automated tool that aligns with the proposed methodology. This tool would enable a valuable comparison with existing source code authorship attribution tools referenced in previous literature.

VIII. DATA AVAILABILITY

The data are available in a publicly accessible GitHub repository at https://github.com/xinyu1118/authorship_attribution/blob/master/40authors.rar

IX. CONFLICTS OF INTEREST

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

REFERENCES

[1] S. G. MacDonell, D. Buckingham, A. R. Gray, and P. J. Sallis, "Software forensics: Extending authorship analysis techniques to computer programs," *J. Law Inf. Sci.*, vol. 13, pp. 34–69, Jan. 2002.

[2] G. Frantzeskou, E. Stamatatos, S. Gritzalis, and S. Katsikas, "Effective identification of source code authors using byte-level information," in *Proc. 28th Int. Conf. Softw. Eng.*, May 2006, pp. 893–896.

[3] A. Neme, J. R. G. Pulido, A. Muñoz, S. Hernández, and T. Dey, "Stylistics analysis and authorship attribution algorithms based on self-organizing maps," *Neurocomputing*, vol. 147, pp. 147–159, Jan. 2015.

[4] M. F. Tennyson and F. J. Mitropoulos, "Choosing a profile length in the SCAP method of source code authorship attribution," in *Proc. IEEE SOUTHEASTCON*, Mar. 2014, pp. 1–6.

[5] M. F. Tennyson, "On improving authorship attribution of source code," in *Proc. Int. Conf. Digit. Forensics Cyber Crime*. Berlin, Germany: Springer, 2012, pp. 58–65.

[6] S. Burrows, A. L. Uitdenbogerd, and A. Turpin, "Comparing techniques for authorship attribution of source code," *Softw., Pract. Exper.*, vol. 44, no. 1, pp. 1–32, Jan. 2014.

[7] S. Swain, G. Mishra, and C. Sindhu, "Recent approaches on authorship attribution techniques—An overview," in *Proc. Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA)*, vol. 1, Apr. 2017, pp. 557–566.

[8] Z. Tian, Q. Zheng, T. Liu, M. Fan, E. Zhuang, and Z. Yang, "Software plagiarism detection with birthmarks based on dynamic key instruction sequences," *IEEE Trans. Softw. Eng.*, vol. 41, no. 12, pp. 1217–1235, Dec. 2015.

[9] S. Burrows, A. L. Uitdenbogerd, and A. Turpin, "Application of information retrieval techniques for source code authorship attribution," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Berlin, Germany: Springer, 2009, pp. 699–713.

[10] T. A. Longstaff and E. E. Schultz, "Beyond preliminary analysis of the WANK and OILZ worms: A case study of malicious code," *Comput. Secur.*, vol. 12, no. 1, pp. 61–77, 1993.

[11] H. Ding and M. H. Samadzadeh, "Extraction of Java program fingerprints for software authorship identification," *J. Syst. Softw.*, vol. 72, no. 1, pp. 49–57, Jun. 2004.

[12] R. C. Lange and S. Mancoridis, "Using code metric histograms and genetic algorithms to perform author identification for software forensics," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.*, Jul. 2007, pp. 2082–2089.

[13] M. Shevertalov, J. Kothari, E. Stehle, and S. Mancoridis, "On the use of discretized source code metrics for author identification," in *Proc. 1st Int. Symp. Search Based Softw. Eng.*, May 2009, pp. 69–78.

[14] W. Wisse and C. Veenman, "Scripting DNA: Identifying the Javascript programmer," *Digit. Invest.*, vol. 15, pp. 61–71, Dec. 2015.

[15] S. I. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2012, pp. 90–94.

[16] C. Boulis and M. Ostendorf, "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams," in *Proc. Int. Workshop Feature Selection Data Mining*, 2005, pp. 9–16.

[17] M. Ali, S. Shiaeles, G. Bendiab, and B. Ghita, "MALGRA: Machine learning and N-gram malware feature extraction and detection system," *Electronics*, vol. 9, no. 11, p. 1777, Oct. 2020.

[18] H. Yao, H. Liu, and P. Zhang, "A novel sentence similarity model with word embedding based on convolutional neural network," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 23, p. e4415, Dec. 2018.

[19] B. Al-Ahmad, A. M. Al-Zoubi, R. Abu Khurma, and I. Aljarah, "An evolutionary fake news detection method for COVID-19 pandemic information," *Symmetry*, vol. 13, no. 6, p. 1091, Jun. 2021.

[20] I. Aljarah, M. Habib, N. Hijazi, H. Faris, R. Qaddoura, B. Hammo, M. Abushariah, and M. Alfawareh, "Intelligent detection of hate speech in Arabic social network: A machine learning approach," *J. Inf. Sci.*, vol. 47, no. 4, pp. 483–501, Aug. 2021.

[21] R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, and M. McConley, "Automated vulnerability detection in source code using deep representation learning," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 757–762.

[22] M. A. Hassonah, R. Al-Sayyed, A. Rodan, A. M. Al-Zoubi, I. Aljarah, and H. Faris, "An efficient hybrid filter and evolutionary wrapper approach for sentiment analysis of various topics on Twitter," *Knowl.-Based Syst.*, vol. 192, Mar. 2020, Art. no. 105353.

[23] A. Hovsepyan, R. Scandariato, W. Joosen, and J. Walden, "Software vulnerability prediction using text analysis techniques," in *Proc. 4th Int. Workshop Secur. Meas. Metrics*, Sep. 2012, pp. 7–10.

[24] B. Wutzl, K. Leibnitz, F. Rattay, M. Kronbichler, M. Murata, and S. M. Golaszewski, "Genetic algorithms for feature selection when classifying severe chronic disorders of consciousness," *PLoS ONE*, vol. 14, no. 7, Jul. 2019, Art. no. e0219683.

[25] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, and L. Wan, "Heterogeneous ensemble for feature drifts in data streams," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springer, 2012, pp. 1–12.

[26] V. Bolón-Canedo, D. Rego-Fernández, D. Peteiro-Barral, A. Alonso-Betanzos, B. Guijarro-Berdiñas, and N. Sánchez-Maroño, "On the scalability of feature selection methods on high-dimensional data," *Knowl. Inf. Syst.*, vol. 56, no. 2, pp. 395–442, Aug. 2018.

[27] K. Himabindu, S. Jyothi, and D. M. Mamatha, "GA-based feature selection for squid's classification," in *Soft Computing and Signal Processing*. Singapore: Springer, 2019, pp. 29–36.

[28] J.-S. Wang and S.-X. Li, "An improved grey wolf optimizer based on differential evolution and elimination mechanism," *Sci. Rep.*, vol. 9, no. 1, pp. 1–21, May 2019.

[29] F. Martinez-Rios and A. Murillo-Suarez, "A multiprocess Salp swarm optimization with a heuristic based on crossing partial solutions," *Proc. Comput. Sci.*, vol. 179, pp. 440–447, Jan. 2021.

[30] H.-L. Zhang, M.-R. Chen, P.-S. Li, and J.-J. Huang, "An improved Harris Hawks optimizer combined with extremal optimization," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 3, pp. 655–682, Mar. 2023.

[31] A. Shih, A. Choi, and A. Darwiche, "A symbolic approach to explaining Bayesian network classifiers," 2018, *arXiv:1805.03364*.

[32] F. A. Siddiky, F. Kabir, and S. M. M. Rahman, "Data generalisation with K-means for scalable data mining," *Int. J. Knowl. Eng. Data Mining*, vol. 2, nos. 2–3, pp. 215–235, 2012.

[33] H. Gweon, M. Schonlau, and S. H. Steiner, "The K conditional nearest neighbor algorithm for classification and class probability estimation," *PeerJ Comput. Sci.*, vol. 5, p. e194, May 2019.

[34] A. R. F. Quiros, R. A. Bedruz, A. C. Uy, A. Abad, A. Bandala, E. P. Dadios, and A. Fernando, "A kNN-based approach for the machine vision of character recognition of license plate numbers," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2017, pp. 1081–1086.

[35] B. I. Al-Ahmad, A. A. Al-Zoubi, M. F. Kabir, M. Al-Tawil, and I. Aljarah, "Swarm intelligence-based model for improving prediction performance of low-expectation teams in educational software engineering projects," *PeerJ Comput. Sci.*, vol. 8, p. e857, Jan. 2022.

[36] S. Ding, X. Xu, and R. Nie, "Extreme learning machine and its applications," *Neural Comput. Appl.*, vol. 25, nos. 3–4, pp. 549–556, Sep. 2014.

[37] A. Caliskan, F. Yamaguchi, E. Dauber, R. Harang, K. Rieck, R. Greenstadt, and A. Narayanan, "When coding style survives compilation: De-anonymizing programmers from executable binaries," 2015, *arXiv:1512.08546*.

[38] X. Yang, G. Xu, Q. Li, Y. Guo, and M. Zhang, "Authorship attribution of source code by using back propagation neural network based on particle swarm optimization," *PLoS ONE*, vol. 12, no. 11, Nov. 2017, Art. no. e0187204.

[39] P. Bora, T. Awalgaonkar, H. Palve, R. Joshi, and P. Goel, "ICodeNet—A hierarchical neural network approach for source code author identification," in *Proc. 13th Int. Conf. Mach. Learn. Comput.*, Feb. 2021, pp. 180–185.

[40] M. Baygin, "Classification of text documents based on Naive Bayes using N-gram features," in *Proc. Int. Conf. Artif. Intell. Data Process. (IDAP)*, Sep. 2018, pp. 1–5.

[41] A. Dogan, M. Akay, P. D. Barua, M. Baygin, S. Dogan, T. Tuncer, A. H. Dogru, and U. R. Acharya, "PrimePatNet87: Prime pattern and tunable q-factor wavelet transform techniques for automated accurate EEG emotion recognition," *Comput. Biol. Med.*, vol. 138, Nov. 2021, Art. no. 104867.

[42] S. Key, M. Baygin, S. Demir, S. Dogan, and T. Tuncer, "Meniscal tear and ACL injury detection model based on AlexNet and iterative ReliefF," *J. Digit. Imag.*, vol. 35, no. 2, pp. 200–212, Apr. 2022.

[43] S. Park, S. Ko, J. Choi, H. Han, S.-J. Cho, and J. Choi, "Detecting source code similarity using code abstraction," in *Proc. 7th Int. Conf. Ubiquitous Inf. Manage. Commun.*, Jan. 2013, pp. 1–9.

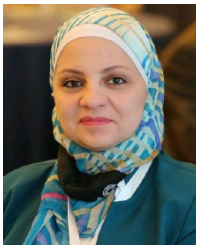
[44] M. Abdolahi and M. Zahedh, "Sentence matrix normalization using most likely N-grams vector," in *Proc. IEEE 4th Int. Conf. Knowl.-Based Eng. Innov. (KBEI)*, Dec. 2017, pp. 0040–0045.

[45] F. Elghannam, "Text representation and classification based on bi-gram alphabet," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 2, pp. 235–242, Feb. 2021.

[46] Y. HaCohen-Kerner, D. Miller, Y. Yigal, and E. Shayovitz, "Cross-domain authorship attribution: Author identification using char sequences, word unigrams, and POS-tags features," in *Proc. Notebook PAN CLEF*, 2018, pp. 1–14.

[47] A. Dhar, H. Mukherjee, S. M. Obaidullah, and K. Roy, "An ensemble learning based author identification system," in *Proc. Int. Conf. Pattern Recognit. Mach. Intell.* Cham, Switzerland: Springer, 2019, pp. 65–73.
[48] M. Garg, "UBIS: Unigram bigram importance score for feature selection from short text," *Expert Syst. Appl.*, vol. 195, Jun. 2022, Art. no. 116563.
[49] R. Jin, A. Hauptmann, and R. Yan, "Image classification using a bigram model," in *Proc. AAAI Spring Symp. Intell. Multimedia Knowl. Manag.*, 2003, pp. 8–15.
[50] R. Nazar and M. S. Pol, "An extremely simple authorship attribution system," in *Proc. 2nd Eur. IAFL Conf. Forensic Linguistics/Language Law*, 2006, pp. 1–7.
[51] X. Kong, "An exploration of source code authorship attribution using sequence learning," Ph.D. thesis, Cooper Union Advancement Sci. Art, New York, NY, USA, 2020.
[52] M. F. Kabir and S. A. Ludwig, "Association rule mining based on ethnic groups and classification using super learning," in *Applied Smart Health Care Informatics: A Computational Intelligence Perspective*. Hoboken, NJ, USA: Wiley, 2022, pp. 111–129.
[53] M. F. Kabir and S. A. Ludwig, "Enhancing the performance of classification using super learning," *Data-Enabled Discovery Appl.*, vol. 3, no. 1, p. 5, Dec. 2019.
[54] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Appl. Soft Comput.*, vol. 62, pp. 441–453, Jan. 2018.
[55] M. Mafarja, I. Aljarah, H. Faris, A. I. Hammouri, A. M. Al-Zoubi, and S. Mirjalili, "Binary grasshopper optimisation algorithm approaches for feature selection problems," *Expert Syst. Appl.*, vol. 117, pp. 267–286, Mar. 2019.
[56] A. Alzaqebah, I. Aljarah, O. Al-Kadi, and R. Damaševičius, "A modified grey wolf optimization algorithm for an intrusion detection system," *Mathematics*, vol. 10, no. 6, p. 999, Mar. 2022.
[57] A. Alzaqebah, I. Aljarah, and O. Al-Kadi, "A hierarchical intrusion detection system based on extreme learning machine and nature-inspired optimization," *Comput. Secur.*, vol. 124, Jan. 2023, Art. no. 102957.
[58] Q. Al-Tashi, S. J. Abdulkadir, H. M. Rais, S. Mirjalili, H. Alhussian, M. G. Ragab, and A. Alqushaibi, "Binary multi-objective grey wolf optimizer for feature selection in classification," *IEEE Access*, vol. 8, pp. 106247–106263, 2020.
[59] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.
[60] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009–2019)," *IEEE Access*, vol. 9, pp. 26766–26791, 2021.
[61] G. Baldini and D. Geneiatakis, "A performance evaluation on distance measures in KNN for mobile malware detection," in *Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2019, pp. 193–198.
[62] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, May 2018.
[63] J. Li, H. Kang, G. Sun, T. Feng, W. Li, W. Zhang, and B. Ji, "IBDA: Improved binary dragonfly algorithm with evolutionary population dynamics and adaptive crossover for feature selection," *IEEE Access*, vol. 8, pp. 108032–108051, 2020.
[64] L.-Y. Chuang, C.-H. Yang, and J.-C. Li, "Chaotic maps based on binary particle swarm optimization for feature selection," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 239–248, Jan. 2011.
[65] A. Thakkar and K. Chaudhari, "Information fusion-based genetic algorithm with long short-term memory for stock price and trend prediction," *Appl. Soft Comput.*, vol. 128, Oct. 2022, Art. no. 109428.
[66] R. M. Math and N. V. Dharwadkar, "Early detection and identification of grape diseases using convolutional neural networks," *J. Plant Diseases Protection*, vol. 129, no. 3, pp. 521–532, Jun. 2022.
[67] H. M. J. Mustafa, M. Ayob, D. Albashish, and S. Abu-Taleb, "Solving text clustering problem using a memetic differential evolution algorithm," *PLoS ONE*, vol. 15, no. 6, Jun. 2020, Art. no. e0232816.
[68] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 856–863.
[69] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2012.
[70] T. M. Cover, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 1999.
[71] I. Kononenko, E. Šimec, and M. Robnik-Šikonja, "Overcoming the Myopia of inductive learning algorithms with RELIEFF," *Appl. Intell.*, vol. 7, no. 1, pp. 39–55, Jan. 1997.
[72] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, Oct. 2017.

BILAL AL-AHMAD received the B.Sc. degree in computer information systems from the Jordan University of Science and Technology (JUST), Jordan, in 2006, the M.Sc. degree in computer information systems from Yarmouk University, Jordan, in 2009, and the Ph.D. degree in software engineering from North Dakota State University, USA, in 2015. He has been an Associate Professor with the Computer Information Systems Department, The University of Jordan, Aqaba, since May 2022. He has been an Associate Professor in software engineering with the CSIT Department, Saint Cloud State University, MN, USA, since August 2023. His research interests include requirements engineering, software design, software testing, software construction, machine learning, optimization, and educational software engineering.



NAILAH AL-MADI received the Bachelor of Science degree (Hons.) from Al Al-Bayt University, the master's degree in computer science from the Jordan University of Science and Technology, and the Ph.D. degree (Hons.) in computer science from North Dakota State University, in 2014. She is currently an Associate Professor in computer science with the Princess Sumaya University for Technology. She has an impressive publication record, having authored over 25 papers in reputable journals and international conferences. Her scholastic excellence has been recognized through numerous awards, including the Best Teacher Award, in 2019, the Doctoral Dissertation Award, in 2013, and a distinction as one of the top 10% of students during the Ph.D. studies, in 2012. In addition to her personal academic accomplishments, she has contributed to the academic community by supervising numerous award-winning projects across various local and international competitions. She has also played a significant role in academic mentorship, supervising, and examining master's students at several universities. Her research interests include computer science, including optimization and evolutionary computation, data mining, artificial intelligence, robotics, and wireless sensor networks. Through her extensive research and supervisory roles, she continues to make a significant impact in advancing knowledge and fostering academic excellence within the community.

ABDULLAH ALZAQEBAH received the B.Sc. degree in computer information systems from the Jordan University of Science and Technology (JUST), Jordan, in 2008, the M.Sc. degree in computer science from Al-Balqa Applied University, Jordan, in 2014, and the Ph.D. degree in computer science from The University of Jordan, Jordan, in 2022. He has been an Assistant Professor in computer science with Al-Ahliyya Amman University, since October 2023. His research interests include machine learning, cybersecurity, data science, optimization, and artificial intelligence.

RAMI S. ALKHAWALDEH received the B.S. degree in computer information systems from Yarmouk University, Irbid, Jordan, in 2007, the M.Sc. degree in computer information systems from The University of Jordan, Amman, Jordan, in 2010, and the Ph.D. degree in computing science from Glasgow University, U.K., in 2017. From 2010 to 2012, he was a Lecturer with The University of Jordan, where he contributed to the education and development of students in the field of computer information systems. Since February 2021, he has been holding the position of an Associate Professor with the Computer Information Systems Department, The University of Jordan, further enriching the academic environment and sharing his expertise with students and colleagues. His research interests include the range of cutting-edge fields, including artificial intelligence, deep and machine learning, information retrieval, VoIP, and wireless networks.

KHALED ALDEBEI received the Ph.D. degree in computer science from the University of Technology Sydney, Australia. He is currently an Assistant Professor with the Information Technology Department, The University of Jordan, Jordan. He has the higher degree by research publication award from the University of Technology Sydney. He has an Oracle Certified Associate (OCA) and Oracle Certified Professional (OCP). He was with the Computer Centre, The University of Jordan, as a Computer Programmer in system analysis, where he was with the Faculty of Information Technology and Systems as a Teaching Assistant. Since 2018, he has been with the Faculty of Information Technology and Systems as an Assistant Professor. He has many programming skills, e.g., Python, Oracle (Form and Report Builder), MATLAB, and Database (SQL, PL-SQL). He has published many articles in high rank journals and international conferences. Currently, he is the Head of the Information Technology Department and the Computer Information Systems Department. His current research interests include machine learning, data mining, text processing, and natural language processing.

MD. FAISAL KABIR (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science and engineering (CSE) from United International University (UIU), Bangladesh, the M.Sc. degree from the University of Bradford, U.K., under the European Union's eLink Scholarship, and the Ph.D. degree in computer science from North Dakota State University (NDSU), ND, USA. His professional journey includes roles as a Lecturer and later as an Assistant Professor with the Department of CSE, UIU, from 2006 to 2014. Furthermore, he took on roles as a Research/Teaching Assistant and an Instructor with NDSU, from 2014 to 2020, while pursuing the Ph.D. degree. Currently, he is an Assistant Professor with the Computer Science Department, Pennsylvania State University Harrisburg, USA. His research interests include data mining, machine learning, explainable AI, data analytics, and federated learning with applications in healthcare, bio-medical, and other multidisciplinary domains.

ISMAIL ALTAHARWA received the B.Sc. degree in computer science and its applications from The Hashemite University, Jordan, in 2005, the M.Sc. degree in computer science (emphasizes in AI techniques especially computational intelligence and evolutionary computations) from Al-Balqa Applied University, Jordan, in 2008, and the Ph.D. degree in computer science and information engineering (emphasized in machine learning techniques and information security) from the National Taiwan University of Science and Technology, Taiwan, in 2014. He is currently an Associate Professor with the Department of Computer Information Systems, The University of Jordan, Aqaba Campus.

MUA'AD ABU-FARAJ received the B.Eng. degree in computer engineering from Mu'tah University, Jordan, in 2004, the M.Sc. degree in computer and network engineering from Sheffield Hallam University, U.K., in 2005, and the M.Sc. and Ph.D. degrees in computer science and engineering from the University of Connecticut, USA, in 2012. He is currently a Professor in computer engineering with The University of Jordan, Jordan. His scholarly pursuits revolve around re-configurable hardware, cryptography, and digital signal processing. He holds memberships with many associations, including IEEE and the Jordan Engineers Association (JEA).

IBRAHIM ALJARAH (Senior Member, IEEE) received the Ph.D. degree in computer science from North Dakota State University, USA, in 2014. He is currently a highly acclaimed Big Data Scientist and an AI Technical Consultant with extensive experience in the field. He is also a Highly Cited Researcher (Clarivate), who has been recognized for his exceptional research influence worldwide, demonstrated by the production of multiple highly-cited papers that rank in the top 1% by citations for field and year in the Web of Science. He is currently a Professor in big data mining and computational intelligence with the Department of Artificial Intelligence, Jordan. Since, he has been making significant contributions to the field of AI. In 2017, he co-founded the Evolutionary and Machine Learning (Evo-ML.com) Research Group, which includes seven Ph.D. and master's students from The University of Jordan and collaborates with many well-regarded and ambitious scholars from different countries across the world. He has won numerous awards and recognitions for his exceptional research, including the UJ Distinguished Researcher Award, in 2022, and the Ali Mango Award for Distinguished Researcher, in 2020. He was also recognized as one of the top researchers publishing in Scopus journals with The University of Jordan, in 2019, 2020, 2021, 2022, and 2023, with over 130 publications in refereed international conferences and prestigious Q1 and Q2 ISI journals with high impact factor and over 14,400 citations and an H-index of 54. In addition, he has ranked amongst the top 2% scientists in the world and one of the top 10 scientists in Jordan in the artificial intelligence and image processing field, as per Stanford University Global Ranking, in 2020, 2021, and 2022. He has also participated in numerous conferences in the field of data mining, machine learning, and big data, such as CEC, GECCO, EVOStar, ECTA, IEEE NABIC, and BIGDATA Congress. He has contributed significantly to several projects in the USA, such as vehicle class detection system (VCDS), pavement analysis via vehicle electronic telemetry (PAVVET), and farm cloud storage system (CSS) projects. His research interests include data mining, machine learning, big data, mapreduce, hadoop, swarm intelligence, and evolutionary computation. With a wealth of experience and numerous accolades to his name, he is also a Highly Sought-After Expert in the field of AI and big data, with a passion for driving innovation and progress in the field.