





Article

Enhanced Feature Selection Using Genetic Algorithm for Machine-Learning-Based Phishing URL Detection

Emre Kocyigit ¹, Mehmet Korkmaz ², Ozgur Koray Sahingoz ^{3,*} and Banu Diri ²

¹ Interdisciplinary Centre for Security, Reliability and Trust (SnT), Université du Luxembourg, 2 Av. de l'Université, Esch-sur-Alzette, 4365 Luxembourg, Luxembourg; emre.kocyigit@uni.lu

² Department of Computer Engineering, Yildiz Technical University, Besiktas, Istanbul 34349, Türkiye; mkorkmazzz@gmail.com (M.K.); diri@yildiz.edu.tr (B.D.)

³ Department of Computer Engineering, Biruni University, Istanbul 34100, Türkiye

* Correspondence: osahingoz@biruni.edu.tr

Abstract: In recent years, the importance of computer security has increased due to the rapid advancement of digital technology, widespread Internet use, and increased sophistication of cyberattacks. Machine learning has gained great interest in securing data systems because it offers the capability of automatically detecting and responding to security threats in real time, which is crucial for maintaining the security of computer systems and protecting data from malicious attacks. This study concentrates on phishing attack detection systems, a prevalent cyber-threat. These systems assess the features of the incoming requests to identify whether they are malicious or not. Although the number of features is increasing in these systems, feature selection has become an essential pre-processing phase that identifies the most important features of a set of available features to prevent overfitting problems, improve model performance, reduce computational cost, and decrease training and execution time. Leveraging genetic algorithms, known for simulating natural selection to identify optimal solutions, we propose a novel feature selection method, based on genetic algorithms and locally optimized, that is applied to a URL-based phishing detection system with machine learning models. Our research demonstrates that the proposed technique offers a promising strategy for improving the performance of machine learning models.

Keywords: feature selection; genetic algorithm; phishing detection



Citation: Kocyigit, E.; Korkmaz, M.; Sahingoz, O.K.; Diri, B. Enhanced Feature Selection Using Genetic Algorithm for Machine-Learning-Based Phishing URL Detection. *Appl. Sci.* **2024**, *14*, 6081. <https://doi.org/10.3390/app14146081>

Academic Editors: Jianquan Liao, Zhanlong Zhang and Peiyu Jiang

Received: 21 May 2024

Revised: 29 June 2024

Accepted: 30 June 2024

Published: 12 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, computer security issues, such as cyberattacks, data intrusions, and other forms of malicious activities, have increased significantly. This trend is influenced by a number of factors, including a growing dependence on technology, an evolving threat landscape, human error, an absence of cybersecurity awareness, increased connectivity, etc. [1]. Phishing is a type of cyber attack in which attackers attempt to trick users into disclosing sensitive information, such as usernames, passwords, and credit card details, by posing as a trustworthy entity [2]. Numerous methods, including email, social media, instant messaging, and SMS, can be used to carry out phishing attacks. These attacks are designed to deceive users into revealing sensitive information, such as login credentials, credit card information, or other personal data, which can then be used for fraudulent purposes. A successful phishing attack can have severe consequences, including financial losses, identity theft, and reputation damage. In addition to all this, phishing attacks can be carried out to deliver malware or ransomware attacks.

The Anti-Phishing Working Group (APWG) conducted research for the 2nd Quarter 2023 Phishing Activity Trends Report, focusing on the increasing rate of phishing attacks [3]. The findings indicate a notable increase over the past four years, with a growth rate exceeding 150% annually, as illustrated in Figure 1. According to Proofpoint's 2023 State of the Phish Report—Phishing Stats and Trends, 84% of the organizations experienced at

least one successful phishing attack. These successful attacks mainly target the private information of companies, leading to increasing financial losses [4].

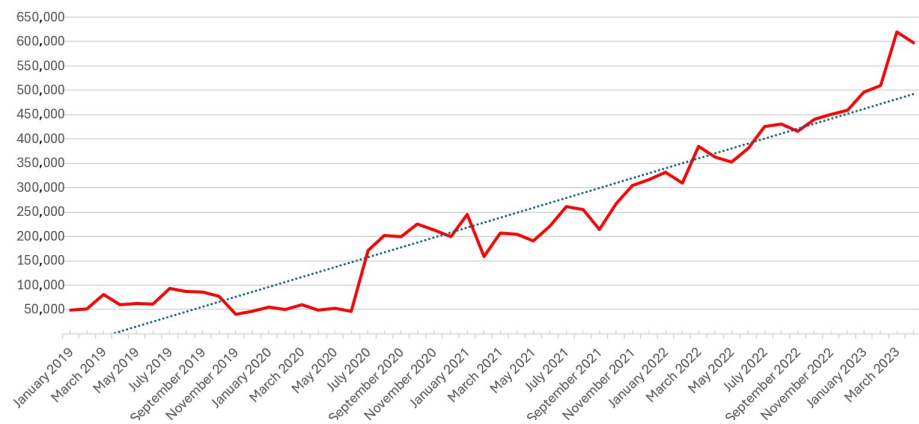


Figure 1. Number of phishing attacks, January 2019 to April 2023 [3].

As seen from this, phishing is still one of the most prevalent and difficult-to-prevent attack types, as attackers constantly change their methods and find ways to circumnavigate security measures. However, there are methods to reduce the risk of phishing attacks, such as educating users and implementing technical controls such as spam filters, URL-based phishing detection [5], content-based phishing detection [6], two-factor authentication [7], etc. The URL-based phishing detection system, which analyzes features derived from URLs such as URL length, use of HTTPS, and domain age, are highly effective in identifying and preventing phishing attacks [8]. This system quickly analyzes and recognizes patterns, making it a popular choice for detecting deceptive URLs that appear legitimate but direct users to fraudulent websites. In recent years, Machine Learning (ML)-based algorithms have gained popularity as a means of protecting computer systems against phishing attacks by automatically identifying malicious traffic. Phishing attacks are difficult to detect and prevent due to the use of sophisticated deception techniques by attackers and the variety of communication channels [9].

Large datasets containing legitimate and phishing can be analyzed using ML algorithms to identify distinguishing patterns and characteristics. These algorithms can use related patterns to identify potential phishing attacks, even if they have never been seen before. Over time, ML can also improve the accuracy and efficiency of phishing detection systems. As new phishing attacks are identified, ML algorithms can update their models to detect future attacks of similar nature.

Big data lead to longer training times in ML algorithms due to the increased computational costs associated with processing larger datasets. Not only the number of samples in the dataset but also its dimensionality affects the time. The high dimensionality of the dataset extends the inference time of ML systems, particularly because of the increased cost of feature extraction. The inference time is vital in real-time phishing detection models because it directly influences users and impairs the run-time performance of the deployed ML models. To deal with this problem, feature reduction, also known as feature selection, is preferred to decrease the computational cost of the ML inference by utilizing a subset of all features. By employing the feature selection methods, it is expected to improve the model's performance metrics, such as accuracy, while reducing the model's complexity and computational cost. Therefore, feature selections are generally preferred to increase the efficiency of ML-based systems.

There are several commonly used feature selection algorithms in the literature, such as Recursive Feature Elimination (RFE) [10], Principal Component Analysis (PCA) [11], selectKBest [12], lasso [13], tree-based methods [14], Evolutionary Algorithms (EAs), etc. EAs [15] are a family of optimization techniques that draw inspiration from the principles of natural evolution. These algorithms use a population of possible solutions that change

over time through a process of selection, mutation, and recombination to find an optimal or near-optimal solution to a problem. They are preferred in many optimization problems, just as they were in feature selection problems, because they are robust, flexible, can handle non-linear and non-differentiable problems, can work in parallel, or can optimize for more than one goal.

Genetic Algorithms (GAs) are evolutionary optimization techniques that try to find the best solution to a problem by imitating how natural selection works. GAs have been used to solve a wide range of optimization problems, such as feature selection, with great success. In this concept, our main challenge fits in a multi-objective optimization problem by increasing the detection accuracy and recall value of the trained system, while decreasing the processing time by minimizing the number of features to use.

The use of GAs for feature selection involves encoding the features into chromosomes and then applying genetic operators such as selection, crossover, and mutation to evolve a population of candidate solutions. Each solution is judged on how well it works with a given objective function, and the process is repeated until a good solution is found.

This paper aims to provide a comprehensive understanding of GA-based feature selection and its potential to improve the performance of ML models by selecting the optimal subset of features and reducing the dimensions of the data. By presenting the latest research in this area, we hope to inspire future work and encourage researchers to explore this promising approach in their applications.

In order to conduct a thorough and efficient systematic study, it is imperative to establish well-defined research questions. The research questions formulated for the present investigation are outlined below:

RQ 1: How can an evolutionary approach (genetic algorithm) be used during the feature selection of machine learning models in the URL-based phishing detection system?

RQ 2: What are the effects of a genetic-algorithm-based feature selection strategy on machine learning models?

RQ 3: What are the strengths and weaknesses of the genetic-algorithm-based feature selection strategy compared to greedy-based feature selection algorithms?

The rest of the paper is organized as follows: In Section 2, related works such as phishing detection, ML, and GA-based systems are detailed by showing the related works in the literature. Section 3 explains the design details and algorithms of the proposed solution for a genetic-algorithm-based feature selection algorithm in a phishing detection system. The empirical results of the proposed system are analyzed by comparison with other feature selection models and are also illustrated in Section 5 after explaining the experimental setup in Section 4. Finally, after discussing the crucial aspects of the genetic-algorithm-based feature selection method in Section 6, the paper is concluded in Section 7 including future work that can be conducted in related areas.

2. Related Work

2.1. Phishing Detection

Phishing detection studies remain important as attacks become more widespread and diversified, and detection studies are also updated and improved with new developments. For example, as ML and Deep Learning (DL) techniques proved themselves with their successful results in various problem fields, researchers in the field of phishing detection began to use these techniques [16]. Most studies focus on phishing websites and emails while employing ML and DL algorithms. In the study by Opara et al. [17], a novel approach is introduced to detect phishing attacks using raw URL and HTML content. The method involves training a dense neural network with corresponding characters, merging embedded matrices of URL and HTML layers, and creating a model of semantic dependencies with convolutional layers. Extensive tests on actual phishing data yielded an accuracy of 98.1%. Another study focusing on phishing websites with recent technology by Adebowale et al. [18] utilized CNN and LSTM network architectures of DL to detect phishing attacks based on the content of the website, such as text and images. ML algorithms,

e.g., K-nearest neighbor, decision Tree, random forest, Ada-boost, XGBoost and Artificial Neural Networks (ANN), are frequently used in phishing detection studies and the overall detection scores are quite high [19].

2.2. Feature Selection

Feature selection, which is the selection of the optimal subset of existing features to improve performance and increase computational efficiency, in ML is a challenging issue due to the complexity of the data. Reducing the dimensions of the data is necessary to reduce this complexity and improve the accuracy of classification. There are different strategies to find the optimal features such as sequential, i.e., greedy hill climbing, exponential, that is, exhaustive, random, and heuristic search among feature selection methods [20]. For example, El-Hasnony et al. propose a new binary variant of the wrapper feature selection algorithm that combines grey wolf optimization and particle swarm optimization in [21]. Heuristic-based feature selection algorithms can be classified as evolution-based, swarm intelligence-based, physics-based, or human-behavior-related, and GA indicated significant performance as discussed in the comprehensive survey of Agrawal et al. [22].

2.3. Genetic-Algorithm-Based Feature Selection

GA was widely used in the feature selection stage of ML pipelines that were implemented to solve various problems such as phishing, spam, malicious domain detection, or disease diagnosis such as cancer detection. Saibene et al. [23] tackled the challenge of heterogeneity and high dimensionality in EEG signals for data interpretation. They proposed a GA for feature selection, modifying stopping criteria and fitness functions to accommodate supervised and unsupervised approaches. The proposed GAFS outperforms benchmarks in overall performance and feature reduction, with consistent features aligned with neuroscientific literature, and the proposal is effective for heterogeneous data. Addressing the high-dimensionality of document representation in text classification tasks, Catak suggests a GA-based meta-heuristic optimization for feature selection [24]. He had developed a new objective function based on the F1score model and the size of the subset of features to improve the F1 score of the classifier hypothesis. In this study, it is shown that the proposed approach can improve the generalization ability of a classifier by reducing redundant and irrelevant features.

Dominguez presents a model with which to recognize and classify children's activities with audio data from embedded sensors in clothing [25]. Due to limited mobile device resources, a GA is used to reduce the size of the dataset, achieving an accuracy of 0.92 with a random forest classifier. In the work of Muhammad Taseer Suleman et al. [26], a model is introduced to detect phishing websites using ML approaches is introduced. They demonstrate improved detection accuracy through feature selection using GA, with the combination of ID3 and YAGGA achieving up to 95% accuracy. The authors proposed a model for the detection of anomalies in breast thermograms using a combination of a transfer-learning-based Deep Learning (DL) model and feature selection approaches [27]. The DL model generates a large number of features and requires a significant amount of memory and computational time to process these features. The number of features is reduced using a hybrid of optimization algorithms GA and graywolf. In tests on an open source dataset, 100% accuracy was achieved using only 3% of the extracted features. In their study, Sekhar and Sujatha [28] used GA for feature selection to improve the classifier's performance and proposed a model called "Subset Generation using Genetic Algorithm for Improved Classification" (ISG-GA-IC). Furthermore, the paper presents the results of studies on the use of EAs for feature extraction and selection in high-dimensional datasets.

Similarly to these, Rostami introduces a GA based on community detection for feature selection, which operates in three steps [29]. First, they calculate feature similarities. Second, community detection algorithms are used to classify features into clusters. Lastly, a GA with a community-based repair operation is used to select features. The performance of the proposed approach is evaluated on nine benchmark classification problems. Fur-

thermore, the performance is compared with three new feature selection methods based on PSO, ACO, and ABC algorithms in three classifiers. The results indicate that the proposed method achieves higher accuracy compared to the PSO, ACO, and ABC algorithms, with average improvements of 0.52%, 1.20%, and 1.57%, respectively. Syed et al. [30] present a wrapper approach for feature selection in multi-target regression. The method utilizes the MTR-SAFER algorithm, a safe semi-supervised regression algorithm designed for multi-target regression with a single target technique. To determine the optimal feature subset, the paper incorporates a GA as the chosen feature selection method. The GA is employed to search for the most suitable set of features that maximize the performance of the MTR-SAFER algorithm.

Microarray datasets are currently being used in the early diagnosis of some chronic diseases, such as cancer. Due to their high dimensionality, reducing dimensions is necessary to increase the accuracy of cancer classification. Ali and Saeed [31] proposed a solution to this problem by first selecting the most significant features of cancer microarray datasets using filter feature selection methods such as information gain, information gain ratio, and Chi-squared. Then, GA was used to further optimize the selected features to improve the accuracy of cancer classification. The early diagnosis of cognitive impairment in Alzheimer's disease is crucial to effective treatment and maintaining independence. Divya and Kumari [32] focus on the classification of Alzheimer's disease, mild cognitive impairment, and normal control based on MRI images from the ADNI dataset. Various feature selection techniques are applied to different classifiers to improve classification performance. They used RFE and GA for feature selection and obtained better results with the latter.

One of the topics that has been recently studied is detecting spam on Twitter proposes a method that simultaneously performs dimensionality reduction and hyperparameter optimization to create a spam prediction model [33]. They have used a modified GA to achieve better performance than Chi2 and PCA feature selection methods, using less than 10% of the total feature space. Detecting spam effectively requires identifying important features that can accurately represent the behavior of spammers. In this regard, Elakkiya and Selvakumar [34] proposed a method that uses GA to combine the selection of feature subsets with multiple evaluation metrics. This approach not only considers the features with the highest weight, but also pays attention to the features with the lowest weight. By comprehensively evaluating all the different evaluation metrics, this method generates an appropriate feature subset that has improved the efficiency of spam detection.

Darwish et al. [35] proposed a model with which to detect malicious domains by passively analyzing DNS data and used a GA to select numerous features of DNS data. They then developed a real-time, accurate, and fast classifier by combining a two-step quantum ant colony optimization (QABC) algorithm with the selected features. In text processing, feature selection is an approach used to reduce dimensions because high dimensionality is a significant problem. Belkarkor et al. [36] used a GA to reduce dimensions in their study and compared it with other filtering methods.

Hybrid and ensemble GAFS approaches are also proposed by researchers. For example, Ali and Ahmed [37] proposed a hybrid intelligent approach for predicting phishing websites using feature selection and weighting methods based on EAs and Deep Neural Networks (DNN). To improve the accuracy of phishing website prediction, they heuristically determined the most effective features and optimal weights of website features with GA. The website features selected and weighted by the GA were then used to train DNN to accurately predict the phishing website. The proposed approach achieved higher accuracy, sensitivity, and specificity in the prediction of phishing websites than other studies. Shreem et al. [38] propose a model that is a hybridization of three algorithms: Binary Genetic Algorithm (BGA), Electromagnetism-like Mechanism (EM), and k-means for educational data analysis. The BGA selection mechanism has been modified on the basis of EM and k-means algorithms. Enhanced BGA has been employed as a wrapper feature selection algorithm to reduce the dimensionality of the data and remove irrelevant or redundant

features, improving the performance of ML classifiers. Wang et al. [39] present the EFS-BGA algorithm, which is an approach to the selection of ensemble characteristics based on a GA. Following the generation of a feature subset by each base feature selector, the EFS-BGA technique employs a GA in order to acquire weights that are optimal for each feature subset. This is in contrast to the conventional GAs, which process individual characteristics in a conventional manner. An integration between GA and ANN is carried out in the work of Mohammed et al. [40] as a pre-processing step. This integration aims to significantly eliminate irrelevant features from the datasets before applying ML techniques. The authors evaluate the performance of the proposed algorithms on five datasets and demonstrate their effectiveness in improving the classification performance compared to other feature selection methods.

Multi-objective problems can be solved using GA-based approaches. For example, in the article by Jasuja, the author presents the use of the diploid genetic algorithm (DGA) for the purpose of multi-objective optimization in the feature selection process [41]. When performing a classification task, the objective is to achieve better performance by removing features that are not relevant. In contrast to conventional feature selection algorithms, which concentrate on either enhancing accuracy or reducing the number of features, the approach described seeks to accomplish both of these goals simultaneously.

3. Methodology

In this paper, we aim to develop a phishing detection system by selecting most significant URL-based features with the help of a genetic algorithm, as mentioned in the flow depicted in Figure 2. Feature selection is a critical step in the ML pipeline for both the model's accuracy and run-time measures such as inference time, i.e., prediction duration of the model. Useless features negatively affect the accuracy of the model and add unnecessary complexity into the model. Moreover, fewer features mean less total feature extraction time, which is critical during prediction in the production environment. Taking into account these aspects, we propose a GAFS strategy which aims to improve the performance scores of phishing detection while reducing the total number of selected features.

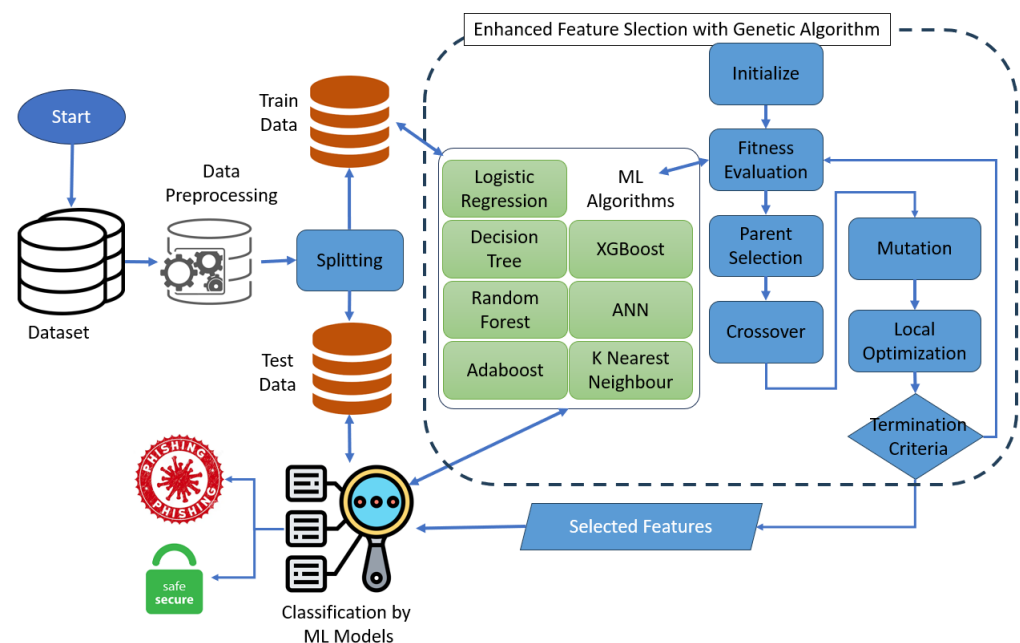


Figure 2. Flowchart of the proposed model.

3.1. Genetic-Algorithm-Based Flow

Finding the best URL-based feature subset among existing features in terms of ML performance measures and minimizing the model's prediction time are objectives of an optimization problem, and searching for all possible solutions is an NP-hard problem. GA is one of the most successful ways of dealing with this type of problem. We decided to use it and adapt our problem into a GA skeleton. Furthermore, we improved it by adding a local optimization step to achieve optimal results in earlier stages of the evolutionary process carried out by the GA. The general flow of the proposed method and GA steps are given in Algorithm 1.

Algorithm 1 Genetic-Algorithm-based Flow

- 1: c represents individual chromosome
 - 2: fv represents fitness value
 - 3: parameters: **pop_size**, **generation**, **mutation_rate**
 - 4: Create initial population using **pop_size** (avoiding duplicates)
 - 5: Calculate fv for each c and store them in local/global tables
 - 6: Transfer best c to the next generation (elitism)
 - 7: **for** $i \leftarrow 1$ to **generation** **do**
 - 8: **Select** two parent c (tournament $k = 4$)
 - 9: **Crossover** the parents and generate a child c (uniform crossover)
 - 10: **Mutate** the child c based on the **mutation_rate** (bit flip)
 - 11: Append the child c to the next generation
 - 12: Apply **local optimization** to randomly selected newly generated c
 - 13: Calculate fv and update the tables
 - 14: **end for**
 - 15: **return** the best c of the last generation
-

3.2. Chromosome

A chromosome serves as a representation of a solution in optimization problems, drawing inspiration from the sequence of genes. In this study, each gene corresponds to a URL-based feature, and its order remains fixed throughout the evolutionary process. The chromosome structure of the proposed model is represented as a binary array with a length of 73, corresponding to the number of URL-based features in our dataset. Each value in the chromosome can be either 1 or 0. A value of 1 indicates the inclusion of the corresponding feature in the training of machine learning models, while a value of 0 indicates its exclusion. For instance, if 10 genes are marked as 1, only these 10 features will be used to train ML-based phishing detection systems, the other features being reduced or extracted.

3.3. Initialization

Initialization in a GA is the process of creating an initial population of potential solutions (individuals or chromosomes) for the optimization problem at hand. The quality and diversity of the initial population can significantly affect the performance and convergence of GA. In some cases, more sophisticated initialization methods, such as using domain-specific knowledge or heuristics, may be employed to improve the performance of GA. In our algorithm, each chromosome is randomly generated while preventing duplicates. This randomness helps us to explore a diverse range of solutions from the beginning. A chromosome must contain at least one "1" to be a valid solution; i.e., it must have at least one feature to train the ML model.

3.4. Fitness Function

The fitness function is a crucial concept that plays a central role in the optimization process. A GA is an optimization algorithm inspired by the process of natural selection and is used to find solutions to complex problems. The fitness function is a key component that evaluates the quality of potential solutions, guiding the algorithm towards better solutions over successive generations.

The fitness function essentially serves as a guide for the algorithm by quantifying how well each potential solution aligns with the desired outcome. The design of an effective fitness function is crucial for the success of a GA. It should accurately reflect the problem's objectives and constraints, providing a clear measure of solution quality. Adjusting the fitness function allows the algorithm to adapt to different domains of problems and optimization goals. The choice of performance metrics depends on the nature of the problem and the goals of the model. While accuracy is a commonly used metric, there are situations where "recall" may be preferred, especially in scenarios where certain types of errors are more critical than others. Detection of positive classes, i.e., phishing attacks, is more critical than overall accuracy, and the accuracy even can be misleading in imbalanced datasets. In this direction, we obtain the "Recall" performance metric as the fitness value of the model in the proposed algorithm. All related calculations were conducted accordingly.

Chromosomes are evaluated according to their fitness values, which indicate how good they are for phishing detection. Fitness values correspond to *recall* scores that are calculated after training the ML models with selected features based on the chromosomes. In other words, an ML model needs to be trained and evaluated once for each chromosome to obtain the fitness value. The training processes differ due to the ML classifiers. For instance, while logistic regression's training time is approximately 1 s with 87 thousand URLs, random forest's training time is over 80 s in the same test machine based on our observations. The total run-time of the evolutionary progress can be too long since the fitness values of all chromosomes of the population need to be calculated for each generation. To reduce this time, we stored the fitness values of each chromosome in a hash table with their keys, which are produced using binary arrays. Then, before calculating the fitness value of a chromosome, we checked the key of the chromosome in the fitness values table so that we prevented repeated calculations of the fitness value.

3.5. Selection Operator

Selection is a crucial step in GA that determines the parent chromosomes, which will be used to produce the offspring chromosome. We prefer the *tournament selection*, which is a flexible and effective model for parent selection in GA and it includes a high level of randomness during selection.

Random chromosomes are selected according to the tournament size parameter (which is selected as 4) in the proposed model parameter, and after competing them, the best one with respect to their fitness values is selected as the parent chromosome. The same process is repeated for the second parent chromosome. For each offspring generation process, two different chromosomes are selected from the existing population.

3.6. Crossover Operator

The crossover operator generates an offspring solution based on the selected chromosomes from the previous generation. Different crossover techniques are available in the literature, such as "one-point crossover", "two-point crossover", and "uniform crossover". We decided to use "uniform crossover" since each gene, i.e., each feature, is independent, and there are no sequential relationships between the genes. A mask with the same size of the parent chromosome is randomly generated for each crossover operation, and "1" means that the related gene of the first parent will be transferred to the offspring, while "0" points out the second parent, as shown in Figure 3.

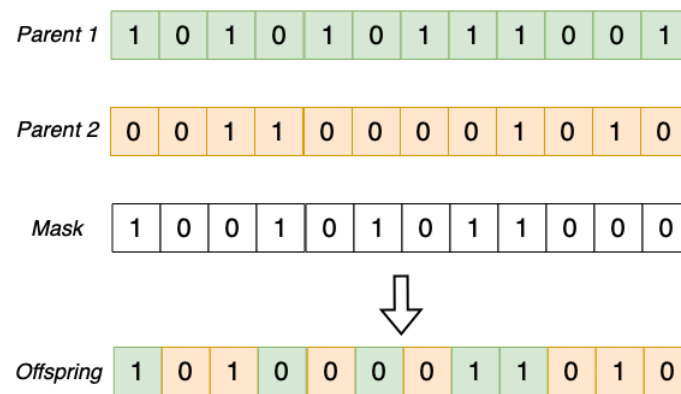


Figure 3. Creating new solution via uniform crossover.

3.7. Mutation Operator

Mutation operators are inspired by the mutation of genes in nature during evolutionary progress, and they increase solution diversity by adding randomness, which helps to pass the local optima. We used “Bit flip mutation” to reach global optimal solutions in which the efficiency depends on the mutation rate.

For example, “0.1” means approximately 10% of each generation will be mutated. Another parameter is flip bit mutation rate, determining the number of genes that will be mutated. “0.1” means 10% of a chromosome, e.g., 8, since the total number of genes in our study is 73, will be flipped. An example is shown in Figure 4.

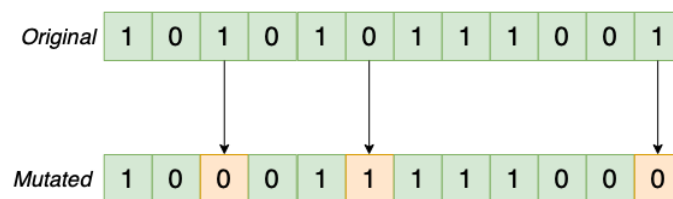


Figure 4. Flip Bit Mutation operator.

3.8. Local Optimization

To quickly achieve better solutions and improve the quality of the solutions, local optimization can be embedded into a GA flow. It also enhances the exploration process by fine-tuning search space and improving individual solutions. We design a local optimization that checks the contributions of genes one by one and updates the existing chromosome in such a way as to increase the performance of ML models.

First, the selected chromosome’s fitness value is obtained. Second, random genes of the chromosome are selected according to a rate, i.e., a parameter. Then, each selected gene is flipped one by one. After each modification, a new fitness value of the modified chromosome is calculated. If the change positively affects performance, we save the modified chromosome. Otherwise, we roll back the modification and cancel the flip operation for that specific gene. Complete steps of the local optimization process for one chromosome are given in Algorithm 2. Since this process increases the cost of overall flow, we did not apply it to all chromosomes but to randomly selected chromosomes based on a parameter.

Algorithm 2 Local Optimization

```

1:  $c$  represents a chromosome
2:  $f_c$  represents the fitness value of the  $c$ 
3: for gene in  $c$  do
4:   Flip the gene
5:   Calculate  $f_c$  of the updated  $c$ 
6:   if gene == 0 then
7:     if new  $f_c <$  old  $f_c$  then
8:       gene = 1
9:     end if
10:  else ▷ gene == 1
11:    if new  $f_c \leq$  old  $f_c$  then
12:      gene = 0
13:    end if
14:  end if
15: end for
16: return the updated  $c$ 

```

3.9. Elitism

In the context of the GA, elitism refers to a strategy in which the best individuals of the current generation are directly passed on to the next generation without undergoing typical genetic operators, such as crossover and mutation. The goal of elitism is to preserve the most fit solutions in the population, ensuring that the best solutions found so far are not lost in the evolution process. Elitism helps maintain a certain level of diversity in the population while ensuring that the best solutions discovered so far continue to contribute to the evolving population. This can be particularly useful in preventing premature convergence to suboptimal solutions and speeding up the convergence towards a high-quality solution.

3.10. Termination Criteria

Termination refers to the conditions that determine when the algorithm should stop its search and return a solution. GA is a type of optimization algorithm inspired by the process of natural selection and genetics. They are used to find approximate solutions to optimization and search problems. Termination criteria are essential to prevent the algorithm from running indefinitely and to define when the algorithm has achieved a satisfactory solution. The choice of termination criteria depends on the specific problem to be solved and the available resources. Common termination criteria for GA are *maximum number of generations*, *convergence*, *fitness threshold*, *sufficient solution*, *computational time limit*, *user-defined criteria*, etc.

It is common to use a combination of these criteria to ensure the termination of the algorithm under various circumstances. Determining appropriate termination criteria is crucial to balance the trade-off between finding a sufficiently good solution and avoiding unnecessary computational costs. In the proposed model, we use *maximum number of generations*.

4. Experimental Setup

Phishing attacks are a major cybersecurity concern due to their widespread use (similar to the DoS, DDos and man-in-the-middle attacks) and high success rate in obtaining sensitive information through social engineering techniques. These attacks exploit human behavior, making them effective despite advanced security measures. Phishing often serves as an entry point for further malicious activities, such as data breaches and ransomware. The financial impact of phishing includes direct theft, fraudulent transactions, and significant mitigation costs. Organizations that fall victim to phishing can suffer severe reputational damage and face regulatory compliance issues, including legal penalties. In addition, attackers frequently exploit current events to make phishing attempts more

convincing, increasing their likelihood of success. Therefore, to detect these attacks, we set a machine-learning-based detection system by using a pre-collected URL dataset, which is defined as high-risk URL dataset, and the performance metrics of the proposed models are detailed in this section.

Seven ML algorithms were selected and experiments were carried out within the scope of the study. Among the most traditional ML algorithms, Logistic Regression (LR), K-Nearest Neighborhood (KNN), Naive Bayes (NB), and Decision Tree (DT) were selected for the experiments. In addition, XGBoost (XGB) and Random Forest (RF) algorithms, which perform ensemble learning, and Artificial Neural Network (ANN) algorithm as a neural network were also used in the experiments.

4.1. Dataset

As of December 2023, there are 1.11 billion websites worldwide, of which 201 million are active [42]. New ones are added to this number every day, some of which are used for phishing attacks. The process of labeling these phishing websites requires a lot of work. There are specific organizations that work on this process, such as <https://phishtank.com/> accessed on 20 May 2024.

The PhishTank.com website, which works to detect and list phishing attacks, works on the basis of tagging URLs by users. Internet users access the PhishTank.com website to query suspicious websites, which members then analyze and tag as “Phishing” or “Legitimate”. The final result of the classification is based on the number of tags. Those websites labeled as “Phishing” or “Legitimate” are added to the list under the respective label. In this way, the black and white lists continue to grow with each passing day of labeling.

This PhishTank.com blacklist is one of the most widely used sources for this and other datasets in the literature, which is classified as phishing on Phishtank.com, can also be found in the datasets in the literature. However, unlike the datasets in the literature, the legitimate websites in this dataset are not obtained from reliable whitelists on the Internet. These websites are also taken from Phishtank.com. For example, in the legitimate class of the dataset, instead of a trusted website such as “www.youtube.com” accessed on 20 May 2024, there are websites with more complex URLs.

The dataset used in the hybrid phishing detection model, proposed by Korkmaz et al. [43], consists of 51,316 legitimate websites and 36,173 phishing websites listed between 2006 and 2021. The highlights of this dataset are shown in Figure 5. This dataset has two main characteristics. First, the data belonging to the legitimate class are websites that members think they are phishing but are legitimate. Therefore, websites belonging to the legitimate class are labeled as risky by us. We have utilized this “High-Risk URL and Content Dataset” in our system because it does not include exact phishing or clean URLs, nor can the URLs be found in white lists such as “www.oracle.com” or “www.nytimes.com” accessed on 20 May 2024. All web pages have a suspicious (unknown) status on the PhishTank website. Although relatively lower scores are expected, the performance metrics are considered more realistic in real-world scenarios.

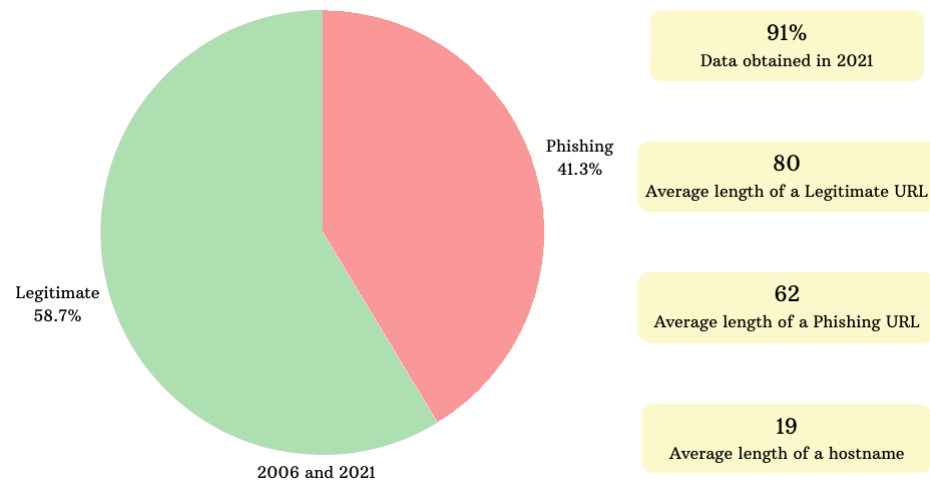


Figure 5. High-risk URL and content dataset details.

4.2. Genetic Algorithm Setting

The impact of predetermined parameters on the performance of the genetic algorithm is of paramount importance as these parameters significantly influence both the resultant outcome and the corresponding computational time required for its attainment. As a result of our preliminary experiments, we decided to use the setting specified in Table 1.

Table 1. Genetic algorithm parameter setting.

Parameter	Setting
Population size	50
Fitness function	Recall of 5-fold cross-validation
Selection method	Tournament
Tournament size	4
Crossover method	Uniform
Mutation method	Flip Bit
Flip Bit rate of selected chromosome's genes	0.07
Mutation rate of each generation	0.1
Local optimization rate	0.01
Termination criterion	50 generations

4.3. Performance Assessment Methods and Testing Parameters

ML models necessitate evaluation through a range of criteria as reliance on a singular metric is inadequate to ensure reliable outcomes. Conventional metrics encompass accuracy, recall, precision, and F-1 score, while in scenarios featuring an imbalanced dataset, the average precision score assumes significance as an indicator of model performance concerning minority dataset. Therefore, we include these metrics in our testing parameters.

The present study aims to identify the optimal set of features through the utilization of a GA-driven optimization technique complemented by local optimization. This selection process not only enhances performance metrics, including accuracy and recall, but also mitigates inference time by significantly reducing the number of features employed. Since each feature extraction operation incurs a computational cost, a reduction in feature usage entails a corresponding reduction in cost. The experimental phase incorporates a carefully monitor of cost fluctuations in relation to the chosen feature list, thereby integrating these changes into the evaluation of performance results.

4.4. Greedy Search Feature Selection Methods

Choosing the right features is essential in ML for constructing effective predictive models. Several greedy search approaches exist, in which features are selected iteratively

based on individual performance, without considering the global optimal subset. There are feature selection approaches, such as filter methods, that rely solely on statistical information of features. Examples of such approaches include RFE [44], Sequential Feature Selection (SFS) [45], Select From Model Feature Selection (SFMFS), which is a technique that selects features according to their importance from a meta-transformer model's perspective [46], and Removing Features with Low Variance (RFLV) with the common filtering method [47], among others.

We applied these feature selection methods and compared their results with our proposed approach. Each method was run on seven selected ML algorithms, and the best results were compared with the results obtained with our proposed feature selection approach. The results are given in Section 5.

5. Results

To evaluate the performance of our proposal method, we first applied popular feature selection methods, which are listed in the previous section, and their recall scores and accuracies are depicted. The results were then compared with the proposed model. Subsequently, the performance scores of the proposed models using the selected features are presented. Given that the proposed evolutionary approach is iteration-based, the evolutionary progress of the method with ML models is demonstrated. The best solutions from the feature selection models select different features; these features are displayed in a comparative analysis.

5.1. Performance Scores of the Greedy Methods

As a popular feature selection model, Removing Features with Low Variance (RFLV) is especially valuable when working with numerical features. At first, it calculates the variance of each numerical feature in the dataset. Variance quantifies the dispersion or spread of data points around the mean. Features with low variance exhibit minimal or insignificant changes in their values, thereby providing less informative input for prediction. Next, we establish a threshold value to determine the minimum acceptable variance below which a feature is deemed to have a low variance. The selection of this threshold can be based on domain knowledge or through experimentation. Within the scope of the experiment, experiments were carried out by selecting four different threshold values as *default*, *0.16*, *0.21*, *0.24*. We eliminated the features that have a lower variance than the specified thresholds. The results of this technique are shown in Table 2.

Table 2. Scores of feature selection with Removing Features With Low Variance.

Versions		V1	V2	V3	V4
Threshold		0	0.16	0.21	0.24
# of Features		72	50	44	37
Logistic	Recall	71.53	72.93	70.27	58.27
Regression	Accuracy	80.11	80.34	79.71	75.56
Random	Recall	87.83	87.59	87.3	86.24
Forest	Accuracy	91.78	91.67	91.47	90.5
XGBoost	Recall	77.50	77.59	77.4	76.07
	Accuracy	85.87	85.66	85.49	83.73
Decision	Recall	85.11	85.43	84.62	83.32
Tree	Accuracy	87.45	87.97	87.52	86.5
Naive	Recall	74.18	74.29	73.55	62.97
Bayes	Accuracy	75.77	76.09	76.30	69.72
KNN	Recall	80.45	80.33	80.19	79.76
	Accuracy	87.11	87.01	86.94	86.45
ANN	Recall	83.58	83.8	84.21	82.81
	Accuracy	88.83	88.82	89.13	88.51

On the other hand, RFE iteratively selects features by recursively considering smaller and smaller feature subsets and selecting the desired number of features of top rank according to their importance. The number of features to be selected for RFE was first tested by determining a wide range. Then, the number of features in the appropriate range was tested by changing the number of features in small steps. When considering the results in Table 3, the highest rates were obtained in the experiments carried out in groups with 45 or 55 features. However, with these results, it can be said that the results obtained in the experiments where random forest and XGBoost algorithms were used are very close to each other.

Table 3. Scores of feature selection with Recursive Feature Elimination.

Versions		V1	V2	V3	V4
# of Features		25	35	45	55
Logistic	Recall	71.75	72.97	74.63	74.33
Regression	Accuracy	80.78	81.6	82.64	82.36
Random	Recall	85.95	87.2	87.71	87.73
Forest	Accuracy	90.65	91.42	91.75	91.72
XGBoost	Recall	84.91	86.32	86.68	87.03
	Accuracy	89.76	90.88	91.06	91.30
Decision	Recall	84.53	84.63	84.79	85.20
Tree	Accuracy	87.14	87.23	87.29	87.60
Naive	Recall	70.64	72.18	72.5	73.16
Bayes	Accuracy	77.25	79.35	78.27	75.85
KNN	Recall	79.01	79.96	80.52	80.34
	Accuracy	86.25	86.78	87.09	86.99
ANN	Recall	81.44	83.39	82.96	83.94
	Accuracy	87.7	88.83	88.56	88.89

Another common feature selection method is SFS, which is a technique that involves systematically adding or removing features from a feature subset to find the optimal set of features for a given ML task. It is a greedy search algorithm that evaluates different feature subsets based on their performance using a specified ML model and a chosen evaluation metric. Within the scope of the study, SFS experiments were performed by selecting 25, 35, 45, and 55 features. In the light of the data shared in Table 4, experiments where random forest and XGBoost algorithms were used are very close to each other in the SFS category. However, it did not give as good results as RFE.

Tree-based classifiers are often used in conjunction with the SFMFS. This is because tree-based models have a built-in mechanism for measuring the importance of each feature in the dataset, which can be used by SFMFS to select the most relevant features. In addition, features are also selected using importance or coefficient attributes according to the threshold value with SFMFS. In this context, experiments were conducted on four different models. Features selected with four different algorithms (decision Tree, logistic regression, random forest and XGBoost) were used in the experiments. With these features, tests were carried out on the previously mentioned selected ML algorithms. As seen in Table 5, the test results performed with the random forest algorithm with 23 features selected using the decision tree algorithm gave the best recall and accuracy rates. These rates are very close to the rates obtained with other feature selection models. What makes it different from other models and thus beneficial is that it uses far fewer features. In experiments with different numbers of feature sets obtained from different feature selection approaches, the SFMFS approach can be preferred when the priority is to reduce the cost through feature selection. However, when considered in terms of recall value, the values obtained with these feature sets need to be improved.

Table 4. Scores of feature selection with Sequential Feature Selection.

Versions		V1	V2	V3	V4
# of Features		25	35	45	55
Logistic	Recall	52.29	51.2	49.95	52.41
Regression	Accuracy	73.12	72.95	73.05	73.84
Random	Recall	79.48	79.33	81.65	87.86
Forest	Accuracy	86.97	86.39	87.85	91.76
XGBoost	Recall	80.66	85.65	85.81	86.36
	Accuracy	87.44	90.48	90.63	90.91
Decision	Recall	75.93	77.79	80.17	83.08
Tree	Accuracy	84.81	85.91	86.62	87.39
Naive	Recall	70.43	70.61	75.16	75.18
Bayes	Accuracy	78.24	78.26	76.63	77.09
KNN	Recall	78.05	79.72	79.37	83.15
	Accuracy	83.23	83.33	84.97	88.18
ANN	Recall	75.71	77.07	77.19	84.79
	Accuracy	84.16	84.98	84.62	89.39

Table 5. Scores of feature selection with SelectFromModel.

Versions		DT-V1	LG-V2	RF-V3	XGB-V4
# of Features		23	22	28	16
Logistic	Recall	71.65	72.16	71.89	72.08
Regression	Accuracy	79.82	81.29	80.06	81.24
Random	Recall	87.04	84.77	86.87	84.86
Forest	Accuracy	91.29	88.85	91.11	88.76
XGBoost	Recall	76.98	74.61	85.28	82.32
	Accuracy	85.34	83.44	90.26	87.73
Decision	Recall	84.18	82.15	83.99	82.42
Tree	Accuracy	86.81	86.71	86.68	86.99
Naive	Recall	69.92	69.33	69.55	72.08
Bayes	Accuracy	75.9	78.33	76.86	79.16
KNN	Recall	79.92	82.27	80.15	82.07
	Accuracy	86.61	86.48	86.76	86.34
ANN	Recall	81.12	79.72	83.36	79.18
	Accuracy	87.5	86.11	88.58	85.69

5.2. Performance Scores of the Proposed Method

We performed our feature selection method with different ML classifiers and obtained the best set of features for each of them. We compared their recall value with the proposed model in Figure 6, and it is clear that the model has a considerable advantage over them. Furthermore, evolutionary progress of each model is shown in Figure 7, which shows the iteration-based enhancement of the proposed model. The best, average, and worst performance scores, i.e., the recall score, of each generation is plotted. Since we utilize elitism in the proposed algorithm, best-score trends always indicate improvement. However, based on average scores, it can be seen that the average success of new generations is sometimes lower than that of previous generations, but general improvement continues. This situation is the natural result of the GA, which includes the randomness and the natural selection principle. We used the same GA parameters to setup the heuristic-based feature selection algorithm, e.g., number of generations, population size, crossover type, mutation type, and local optimization ratio. Then, for each best feature list, we obtained their accuracy, precision, and recall scores, which are shown in Table 6. Although the most important measure for the phishing detection success is the recall score of the positive class, we also

measured fundamental classification metrics such as accuracy and precision in order to better evaluate the overall success of the model. We observed that while maximizing the recall score, the other metrics were also at a reasonable level. Another important result is that our feature selection method improved the performance scores of each classifier compared to the case where all features were used.

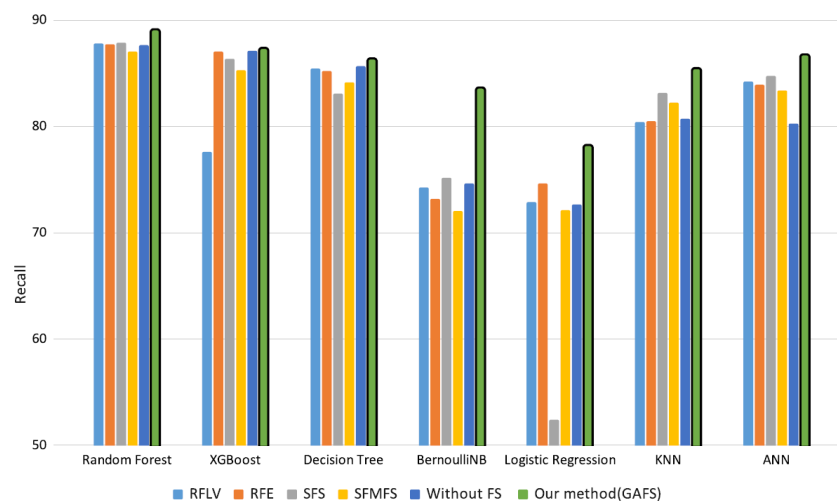


Figure 6. Comparison of feature selection methods with ML models.

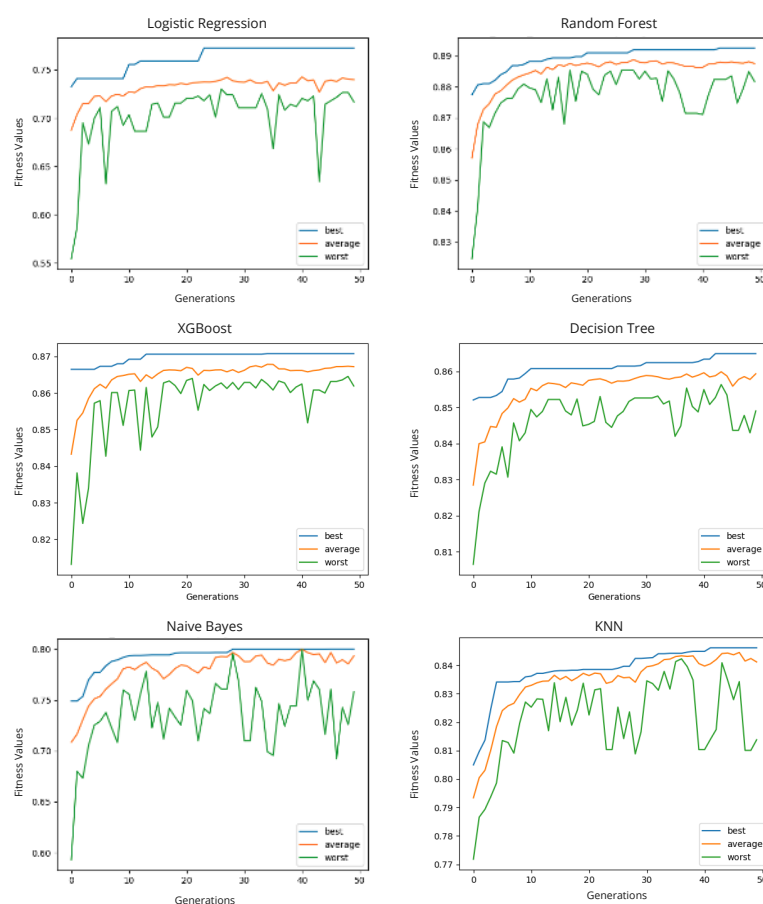


Figure 7. Evolutionary progress of the proposed method with ML models.

Finally, run-time efficiency is also an important performance metric for real-time phishing detection systems. In the proposed model, there are 73 features that can be used for training. Each feature needs to be derived from the URL addresses, making processing time a crucial factor in measuring the system's runtime efficiency. The features in the dataset have extraction times ranging from 0.93 milliseconds (with feature "Does it contain the '@' character?") to 381.08 milliseconds (with feature "Is a well-known brand name used in the URL?"-depending on the number of brand names), with an average of approximately 13.79 milliseconds. This indicates a significant variation in processing time and runtime efficiency. When using the GA, these times can be incorporated into the fitness function calculation. This approach highlights a distinctive aspect of GA-based feature selection models.

Table 6. Performance scores of different models with selected features.

Method	Accuracy	Precision	Recall
Random Forest	92.93 ± 0.71	93.45 ± 0.69	89.05 ± 1.22
XGBoost	91.53 ± 0.36	91.96 ± 0.58	87.24 ± 0.34
Log. Reg.	81.29 ± 0.72	78.17 ± 0.62	76.95 ± 0.53
KNN	89.77 ± 0.60	90.01 ± 0.39	84.65 ± 0.47
Naive Bayes	85.40 ± 0.48	84.12 ± 0.49	80.90 ± 1.06
Decision Tree	88.83 ± 0.59	86.31 ± 0.52	86.64 ± 0.37
ANN	90.07 ± 0.36	89.07 ± 0.61	86.56 ± 0.28

In executing machine learning models with a GA, certain values are determined as constants, as shown in Table 1 based on previous tests. First, the population size, a critical parameter, is set at 50. Although increasing this number could lead to earlier convergence to the optimal value, our goal is to achieve evolutionary improvement based on the iteration count. Second, for iterative enhancement, the iteration count is fixed at 50. As illustrated in Figure 7, the fitness values converge, indicating that there is no need to increase the number of iterations.

5.3. Comparison of Our Proposal and Other Methods

For each ML classifier, we compared the performance of our proposal with other feature selection methods, i.e., RFLV, RFE, SFS, and SFMFS, respectively, which are detailed in Section 5.1. We also checked the performance of the models without any feature selection method, which means that we used all features in training. The results, which are described in Figure 6, prove that our method provides better scores than all others. Due to the fact that all feature selection methods reduced the features, they increased the efficiency in terms of training time and total inference time. For instance, the total inference time with 73 features, that is, all features, is 1.0066 s, and our feature selection method selected 44 features and reduced the feature extraction time to 0.6380 s while increasing performance metrics such as the recall score. These scores are obtained in the Google Colab environment, which has a Tesla T4 GPU-NVIDIA-SMI. This decrease in cost, 36.62%, is critical because it will affect the total prediction time of the model.

Additionally, we compared the best features obtained with each method and our method to understand the relationships between them. The first observation is that 10 of the 73 features are selected by all. Second, our method selected 12 features that were not selected by others while having the best phishing detection scores. The number of features selected by only one method is 1, 1, 2, and 0 for RFLV, RFE, SFS, and SFMFS, respectively. This shows that our method finds 12 different features that could not be discovered by other methods and increases the classification success.

Finally, as illustrated in Figure 8, there is a notable convergence among five different feature selection algorithms, as evidenced by the selection of 10 common features. These features include criteria such as "Is domain name IP address?", "Number of hyphens in domain name", and "Number of digits in the host name field". Such a convergence is

expected as algorithms typically exhibit some degree of similarity in feature identification. However, our GAFS approach diverges from this pattern, identifying 12 additional unique features not highlighted by the other algorithms. These include features like “Is ‘username’ used in the URL?”, “Is there a ‘query’ field in the URL?”, and “Is a well-known brand name used in the URL?”. This distinction underscores the ability of feature selection algorithms to identify the unique characteristics of individual features. Although some features, like “Is a well-known brand name used in the URL?”, cannot be used in real time due to processing costs, the proposed models demonstrate that the GA can select different features compared to other feature selection models. Furthermore, the results suggest that the proposed GAFS approach we used has the ability to discern the collective distinctiveness of these features as a group, rather than just individually. This indicates that the GA approach is adept at capturing a broader and more nuanced spectrum of features, which may be critical in applications where understanding the interplay and collective significance of various features is crucial.

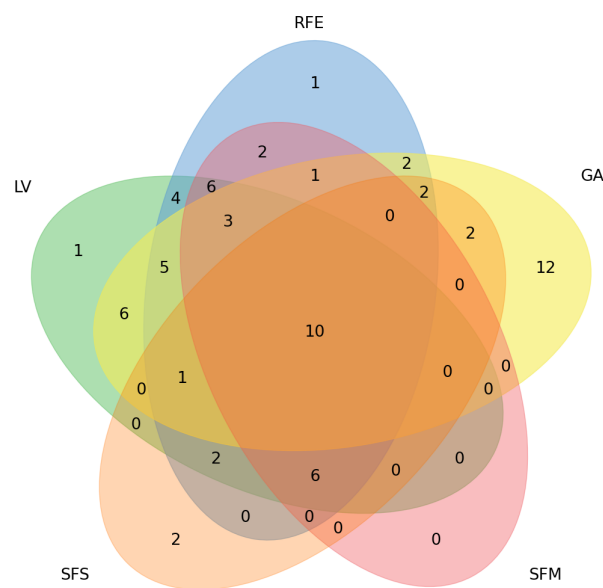


Figure 8. Comparison of selected features by different feature selection methods.

6. Discussion

Feature engineering techniques are crucial in the ML pipeline due to their impact on model's performance, and one of the most common techniques is feature selection that aims to find an optimal subset from the original feature set. Although model-agnostic techniques such as the “Low Variance” method are less expensive in terms of computational cost than modeling-dependent methods, our experiments indicated that the proposed GAFS produced better results.

In the context of the ML pipeline, training the model is the most time-consuming phase after data acquisition and structuring. Specifically, in a GA-based feature selection model, this phase is a dominant part of the training process. The use of GA on the Nvidia-CUDA platform is particularly suitable for master–slave parallelism. This approach involves a master process that manages the population and several slave processes that handle the evaluation of individuals. The master process distributes individuals to the slaves, collects the results, and performs genetic operations. The proposed method is well-suited for parallel computing, unlike other methods such as sequential feature selection. In our approach, each solution within a population and its associated model training can be generated concurrently, presenting a significant advantage of the GA. In an environment with parallel computing infrastructure, the classification performance scores achieved by our method can be enhanced by expanding the search space, such as by increasing

parameters like population size and the number of generations while reducing the total processing time.

Other feature selection methods typically consider the correlation of the features with the target variable. In contrast, the proposed model has adaptability and flexibility to shape the fitness function to meet new requirements or constraints that may emerge such as multi-objective optimization based on fitness values. For example, since the focus of this problem is on phishing detection, the recall value for the positive class is of the utmost importance. However, in the post-deployment phase, if a need arises to limit the total feature extraction time for an input to reduce the application decision time, the same method can be used simply by adjusting the structure of the fitness function. Although other methods may have functions such as specifying a minimum number of features, they do not account for the execution or processing time of the features as a flexible customization capability.

As mentioned previously, the timing of feature calculation is not currently considered in this work. However, the structure of GA is suitable for making selections and taking timing into account. For this calculation, two main metrics are considered: recall value and the calculation time of the feature subset. In our current work, we focus on the recall value as our main fitness value, as depicted in Equation (1). To incorporate feature calculation time, we can modify this fitness function as shown in Equation (2), where a and b are constants that represent the weights of these metrics in the fitness value calculation.

$$\text{FitnessValue} = \text{Recall} \quad (1)$$

$$\text{FitnessValue} = a * \text{Recall} + b * \text{FeatureCalculationTime} \quad (2)$$

This study shows that a powerful optimization algorithm such as a GA has potential in many respects, such as improving ML applications and increasing cost efficiency. Especially with multicore GPUs, the parallel-computing-friendly structure of the GA can be useful not only for feature selection but also for many other problems, such as hyperparameter optimization in the ML and DL models.

7. Conclusions and Future Work

This research introduced a feature selection approach founded upon GA principles, augmented by localized optimization techniques, with the specific objective of classifying phishing websites instead of exhaustively using the entire set of 73 URL-based features. The application of our proposed methodology led to significant improvements in classical performance indicators for ML models, encompassing accuracy, recall, and precision. Furthermore, it is noteworthy that our approach consistently yielded feature subsets comprising fewer than 45 elements across all models, resulting in a substantial reduction in inference-related computational costs.

Although the current study utilized URL-based features, our future work aims to enhance the feature set's versatility by incorporating content-based features such as HTML and CSS attributes. This expansion is intended to improve the effectiveness of phishing attack detection. Additionally, we plan to transform our research into a multi-objective optimization problem, taking into account the phishing detection scores of the model (e.g., accuracy and recall) alongside feature extraction time, which directly impacts costs, especially considering that content-based features are more computationally intensive than URL-based features. As part of our forthcoming efforts, we intend to develop a multi-objective feature selection methodology that adequately addresses these time-related complexities. Additionally, an extension of this research could involve applying the same algorithm to other related optimization problems, such as hyperparameter tuning for ML/DL models using GA. The process of finding the best hyperparameter combinations presents computational challenges similar to the feature selection task, making the proposed algorithm potentially valuable in this context as well.

Author Contributions: Conceptualization, E.K. and O.K.S.; Methodology, M.K. and B.D.; Software, E.K. and M.K.; Resources, E.K.; Data curation, M.K.; Writing—review & editing, B.D.; Supervision, O.K.S.; Project administration, B.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: The data presented in this study are openly available in <https://www.kaggle.com/datasets/mehmetkorkmaz/high-risk-url-and-content-dataset> (accessed on 20 May 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sun, N.; Ding, M.; Jiang, J.; Xu, W.; Mo, X.; Tai, Y.; Zhang, J. Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 1748–1774. [CrossRef]
2. Cloudflare. 2023 Phishing Threats Report. 2023. Available online: <https://www.cloudflare.com/lp/2023-phishing-report/> (accessed on 15 January 2024).
3. APWG. Phishing Activity Trends Report 4th Quarter 2022. 2023. Available online: https://docs.apwg.org/reports/apwg_trends_report_q4_2022.pdf (accessed on 15 January 2024).
4. Proofpoint. 2023 State of the Phish Report—Phishing Stats and Trends. 2023. Available online: <https://www.proofpoint.com/us/blog/security-awareness-training/2023-state-of-the-phish-findings-sneak-peek> (accessed on 15 January 2024).
5. Karim, A.; Shahroz, M.; Mustofa, K.; Belhaouari, S.B.; Joga, S.R.K. Phishing Detection System Through Hybrid Machine Learning Based on URL. *IEEE Access* **2023**, *11*, 36805–36822. [CrossRef]
6. Ma, Y.; Jiang, Z.; Jiang, J.; Zhang, K.; Ling, Z.; Yang, P. Phishsifter: An Enhanced Phishing Pages Detection Method Based on the Relevance of Content and Domain. In Proceedings of the 2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Rio de Janeiro, Brazil, 24–26 May 2023; pp. 909–916. [CrossRef]
7. Sun, Y.; Zhu, S.; Zhao, Y.; Sun, P. A User-Friendly Two-Factor Authentication Method against Real-Time Phishing Attacks. In Proceedings of the 2022 IEEE Conference on Communications and Network Security (CNS), Austin, TX, USA, 3–5 October 2022; pp. 91–99. [CrossRef]
8. Sahingoz, O.K.; Buber, E.; Demir, O.; Diri, B. Machine learning based phishing detection from URLs. *Expert Syst. Appl.* **2019**, *117*, 345–357. [CrossRef]
9. Safi, A.; Singh, S. A systematic literature review on phishing website detection techniques. *J. King Saud Univ.—Comput. Inf. Sci.* **2023**, *35*, 590–611. [CrossRef]
10. Jeon, H.; Oh, S. Hybrid-Recursive Feature Elimination for Efficient Feature Selection. *Appl. Sci.* **2020**, *10*, 3211. [CrossRef]
11. Vidal, R.; Ma, Y.; Sastry, S. Generalized principal component analysis (GPCA). *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1945–1959. [CrossRef] [PubMed]
12. Gupta, K. Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning. 2020. Available online: <https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48> (accessed on 17 June 2024).
13. Li, F.; Lai, L.; Cui, S. On the Adversarial Robustness of LASSO Based Feature Selection. *IEEE Trans. Signal Process.* **2021**, *69*, 5555–5567. [CrossRef]
14. Shobana, G.; Bushra, S.N. Classification of Myopia in Children using Machine Learning Models with Tree Based Feature Selection. In Proceedings of the 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 5–7 November 2020; pp. 1599–1605. [CrossRef]
15. Rey, C.C.T.; García, V.S.; Villuendas-Rey, Y. Evolutionary feature selection for imbalanced data. In Proceedings of the 2023 Mexican International Conference on Computer Science (ENC), Guanajuato, Mexico, 11–13 September 2023; pp. 1–7. [CrossRef]
16. Catal, C.; Giray, G.; Tekinerdogan, B.; Kumar, S.; Shukla, S. Applications of deep learning for phishing detection: A systematic literature review. *Knowl. Inf. Syst.* **2022**, *64*, 1457–1500. [CrossRef] [PubMed]
17. Opara, C.; Chen, Y.; Wei, B. Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. *Expert Syst. Appl.* **2024**, *236*, 121183. [CrossRef]
18. Adebawale, M.A.; Lwin, K.T.; Hossain, M.A. Intelligent phishing detection scheme using deep learning algorithms. *J. Enterp. Inf. Manag.* **2023**, *36*, 747–766. [CrossRef]
19. Shahrivari, V.; Darabi, M.M.; Izadi, M. Phishing Detection Using Machine Learning Techniques. *arXiv* **2020**, arXiv:2009.11116. <http://arxiv.org/abs/2009.11116>.
20. Venkatesh, B.; Anuradha, J. A review of feature selection and its methods. *Cybern. Inf. Technol.* **2019**, *19*, 3–26. [CrossRef]
21. El-Hasnony, I.M.; Barakat, S.I.; Elhoseny, M.; Mostafa, R.R. Improved feature selection model for big data analytics. *IEEE Access* **2020**, *8*, 66989–67004. [CrossRef]

22. Agrawal, P.; Abutarboush, H.F.; Ganesh, T.; Mohamed, A.W. Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009–2019). *IEEE Access* **2021**, *9*, 26766–26791. [\[CrossRef\]](#)
23. Saibene, A.; Gasparini, F. Genetic algorithm for feature selection of EEG heterogeneous data. *Expert Syst. Appl.* **2023**, *217*, 119488. [\[CrossRef\]](#)
24. Catak, F.O. Genetic algorithm based feature selection in high dimensional text dataset classification. *WSEAS Trans. Inf. Sci. Appl.* **2015**, *12*, 290–296.
25. García-Domínguez, A.; Galván-Tejada, C.E.; Zanella-Calzada, L.A.; Rosales, H.G.; Galván-Tejada, J.I.; Celaya-Padilla, J.M.; Luna-García, H.; Magallanes-Quintanar, R. Feature selection using genetic algorithms for the generation of a recognition and classification of children activities model using environmental sound. *Mob. Inf. Syst.* **2020**, *2020*, 8617430:1–8617430:12. [\[CrossRef\]](#)
26. Suleman, M.T.; Awan, S.M. Optimization of URL-based phishing websites detection through genetic algorithms. *Autom. Control Comput. Sci.* **2019**, *53*, 333–341. [\[CrossRef\]](#)
27. Pramanik, R.; Pramanik, P.; Sarkar, R. Breast cancer detection in thermograms using a hybrid of GA and GWO. *Expert Syst. Appl.* **2023**, *219*, 119643. [\[CrossRef\]](#)
28. Sekhar, P.R.; Sujatha, B. Feature extraction and independent subset generation using genetic algorithm for improved classification. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 503–512.
29. Rostami, M.; Berahmand, K.; Forouzandeh, S. A novel community detection based genetic algorithm for feature selection. *J. Big Data* **2021**, *8*. [\[CrossRef\]](#)
30. Syed, F.H.; Tahir, M.A.; Rafi, M.; Shahab, M.D. Feature selection for semi-supervised multi-target regression using genetic algorithm. *Appl. Intell.* **2021**, *51*, 8961–8984. [\[CrossRef\]](#)
31. Ali, W.; Saeed, F. Hybrid filter and genetic algorithm-based feature selection for improving cancer classification in high-dimensional microarray data. *Processes* **2023**, *11*, 562. [\[CrossRef\]](#)
32. Divya, R.; Kumari, R.S.S. Genetic algorithm with logistic regression feature selection for Alzheimer’s disease classification. *Neural Comput. Appl.* **2021**, *33*, 8435–8444. [\[CrossRef\]](#)
33. Ghatasheh, N.; Altaharwa, I.; Aldebei, K. Modified genetic algorithm for feature selection and hyper parameter optimization: Case of XGBoost in spam prediction. *IEEE Access* **2022**, *10*, 84365–84383. [\[CrossRef\]](#)
34. Elakkiya, E.; Selvakumar, S. GAMEFEST: Genetic algorithmic multi evaluation measure based feature selection technique for social network spam detection. *Multimed. Tools Appl.* **2020**, *79*, 7193–7225. [\[CrossRef\]](#)
35. Darwish, S.M.; Farhan, D.A.; Elzoghbi, A.A. Building an effective classifier for phishing web pages detection: A quantum-inspired biomimetic paradigm suitable for big data analytics of cyber attacks. *Biomimetics* **2023**, *8*, 197. [\[CrossRef\]](#)
36. Belkarkor, S.; Hafidi, I.; Nachaoui, M. Feature selection for text classification using genetic algorithm. In Proceedings of the Advances in Machine Intelligence and Computer Science Applications, Khouribga, Morocco, 28–29 November 2022; Aboutabit, N., Lazaar, M., Hafidi, I., Eds.; Springer: Cham, Switzerland, 2023; pp. 69–80.
37. Ali, W.; Ahmed, A.A. Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting. *IET Inf. Secur.* **2019**, *13*, 659–669. [\[CrossRef\]](#)
38. Shreem, S.S.; Turabieh, H.; Azwari, S.A.; Baothman, F. Enhanced binary genetic algorithm as a feature selection to predict student performance. *Soft Comput.* **2022**, *26*, 1811–1823. [\[CrossRef\]](#)
39. Wang, H.; He, C.; Li, Z. A new ensemble feature selection approach based on genetic algorithm. *Soft Comput.* **2020**, *24*, 15811–15820. [\[CrossRef\]](#)
40. Mohammed, T.A.; Bayat, O.; Uçan, O.N.; Alhayali, S. Hybrid efficient genetic algorithm for big data feature selection problems. *Found. Sci.* **2020**, *25*, 1009–1025. [\[CrossRef\]](#)
41. Jasuja, A. Feature selection using diploid genetic algorithm. *Ann. Data Sci.* **2020**, *7*, 33–43. [\[CrossRef\]](#)
42. Huss, N. How Many Websites Are There in the World? 2024. Available online: <https://themeisle.com/blog/how-many-websites-are-there/#gref> (accessed on 15 January 2024).
43. Korkmaz, M.; Kocyigit, E.; Sahingoz, O.K.; Diri, B. A Hybrid Phishing Detection System by Using Deep Learning-based URL and Content Analysis. *Elektron. Ir Elektrotehnika* **2022**, *28*, 80–89. [\[CrossRef\]](#)
44. Chen, X.w.; Jeong, J.C. Enhanced recursive feature elimination. In Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA 2007), Cincinnati, OH, USA, 13–15 December 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 429–435.
45. Rückstieß, T.; Osendorfer, C.; Van Der Smagt, P. Sequential feature selection for classification. In Proceedings of the AI 2011: Advances in Artificial Intelligence: 24th Australasian Joint Conference, Perth, Australia, 5–8 December 2011; Proceedings 24; Springer: Berlin/Heidelberg, Germany, 2011; pp. 132–141.
46. SelectFromModel—scikit-learn.org. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html#sklearn.feature_selection.SelectFromModel (accessed on 19 June 2024).
47. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.