NOVA IMS Information Management School

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

Phishing Website Detection

Phishing website detection using genetic algorithm-based feature selection and parameter hypertuning

Ana Sofia Pulquério Silva

Dissertation

presented as partial requirement for obtaining the Master Degree Program in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

PHISHING WEBSITE DETECTION

by

Ana Sofia Pulquério Silva

Dissertation presented as a partial requirement for obtaining the master's degree in Advanced Analytics

Supervisor: Prof. Roberto Henriques, Ph.D.

February 2023

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledge the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Setúbal, January 25th 2023

DEDICATION

For my parents, of whom I am so proud, and who have always supported and motivated me.

For my sister, grandparents, and grandmother Ilda, *in memoriam*,
who have always accompanied me and whom I miss so much.

ACKNOWLEDGEMENTS

I would like to acknowledge and give my warmest thanks to my supervisor Prof. Roberto Henriques who made this work possible. I would also like to thank Prof. Isabel Abreu dos Santos for her advice and support through all the stages of writing my dissertation and for her brilliant comments and suggestions.

I would also like to thank my partner Filipe Rovisco and my family for their continuous support and understanding when undertaking my research and writing my dissertation.

ABSTRACT

False webpages are created by cyber attackers who seek to mislead users into revealing sensitive and personal information, from credit card details to passwords. Phishing is a class of cyber attacks that mislead users into clicking on false websites, logging into related accounts, and subsequently stealing funds. This cyberattack increases annually given the exponential increase of e-commerce customers, which causes difficulty to distinguish between harmless and false websites. The conventional methods to detect phishing websites are focused on a database of blacklisted and whitelisted. Such methods are not capable to detect new phishing websites. To solve this problem, researchers are developing machine learning (ML) and deep learning-based methods. In this dissertation, a hybrid-based solution, which uses genetic algorithms and ML algorithms for phishing detection based on the URL of the website is proposed. Regarding evaluation, comparisons between conventional ML and DL models are performed using various feature sets resulting from commonly used feature selection methods, such as mutual information and recursive feature elimination. This dissertation proposes a final model with an accuracy of 95.34% on the test set.

KEYWORDS

Phishing; Artificial Intelligence; Machine Learning; Deep Learning; Evolutionary Algorithms; Genetic Algorithms

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS AND ACRONYMS

IoT	Internet-of-Things
DoS	Denial of service
VoIP	Voice over IP
URL	Uniform Resource Locator
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
EA	Evolutionary Algorithms
GA	Genetic Algorithm
MI	Mutual Information
RFE	Recursive Feature Elimination
RF	Random Forest
XGBoost	Extreme Gradient Boosting
KNN	K-nearest neighbors
NB	Naïve Bayes
DNN	Deep Neural Network
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
HL	Hybrid Learning
ID3	Iterative Dichotomiser-3
CNN	Convolutional Neural Network
CRISP-DM	Cross Industry Standard Process for Data Mining

LR	Logistic Regression
SVM	Support Vector Machine
MNB	Multinomial Naïve Bayes
LR	Logistic Regression
ANN	Artificial Neural Network
T-SNE	T-Distributed Stochastic Neighbor Embedding

1. INTRODUCTION

With the advanced developments of computer networks and cloud technology services, significant growth of electronic and mobile usage has been felt, with people increasingly sharing their personal information online, such as bank account details, passwords, and account credentials. The reason behind the exponential increase in security breaches can be attributed to the reliance on digitalization and the Internet of Things (IoT). In the past few years, security incidents such as unauthorized access, malware attack, zero-day attack, data breach, denial of service (DoS), social engineering, and phishing, have been increasing dramatically over the last years.

Phishing is a fraudulent process, where an attacker tries to acquire personal information from their target. These attacks are made via several sources (emails, text messages, or websites) (Vrbančič, Fister, & Podgorelec, 2020). Phishing attacks have become more common and sophisticated, resulting in a significant increase in sophistication and frequency. Attackers are using different channels to trick users, which can be social networks or Voice over IP (VoIP). These channels represent various types of threats, such as malicious email attachments, mobile messages, scam calls, and other types (Alkhalil, Hewage, Nawaf, & Khan, 2021).

To prevent users' information from being compromised and to target unreliable sources, the first available approaches are list-based solutions that collect valid and false Uniform Verified Phishing Resource Locators (URL) for a white or black list. According to L. Tang & Mahmoud (2021), these methods efficiently avoid the reuse of the same URL market as phishing, decreasing the number of users affected and their losses. However, these approaches have one significant disadvantage: the inability to spot new phishing websites. As such, high-accuracy prediction of phishing sites must be addressed with new and innovative capabilities.

Furthermore, it is essential to create advanced tools and technologies to help detect, investigate, and make faster decisions for emerging threats. The significant growth of Artificial Intelligence (AI) has been felt in the past few years in the context of data analytics and Machine Learning (ML), which enables applications to function intelligently. ML typically allows a system to learn and improve through experience, permitting applications to operate efficiently and make intelligent decisions. Despite the numerous AI-based contributions to this subject, solutions combining machine learning and deep learning (DL) have shown promise in website phishing detection using URLs, as they can handle large volumes of data and a wide range of data attributes used for classification.

Due to the large volumes of data and attributes, a spam detector should be able to achieve scalability and interpretability. The selection of features in a data set is critical in affecting the performance of classifiers. The conventional feature selection methods used in the literature to retrieve them cannot effectively target the most appropriate website features for all datasets. The importance of exploring new methodologies regarding feature selection is seen as a promising solution by Ali & Ahmed (2019), Yang, Zhao, & Zeng (2019) and Yang et al. (2019). As it was exposed by (Iuga, Nurse, & Erola, 2016a), the rule-based feature selection and modeling has a limitation in the generalization performance for unobserved URLs. To address this issue, ML and DL algorithms were actively studied. This encourages this dissertation to apply evolutionary algorithms (EA)-based feature selection to enhance phishing website detection.

EAs are based on the Theory of Evolution by Charles Darwin and are distinguished by their specific operators such as recombination (or crossover), mutation, and selection - which can be applied in each cycle or with a probability, whereas each consecutive cycle is defined as a generation. In an EA, a population is initialized with random candidate solutions and these are evaluated. Then, until the priori-defined termination conditions are met, the population is randomly selected. Finally, when the termination conditions are satisfied, it returns the best candidates found. When the termination conditions are satisfied, it returns the best solution found. Being one type of EA, the "genetic algorithm (GA) is a problem-solving method that uses genetics as its model of problem-solving. It's a search technique to find approximate solutions to optimization and search problems." (Sivanandam & Deepa, 2010)

Regarding search problems and optimization, as Ali et al. (2019) referred to, "parameter tuning is an unavoidable task regardless of the nature of the underlying prediction model". Optimization starts with initial values for the ML models' parameters, and because these values may not be the best ones to use, there is a need to change until the best solution is achieved. The importance of optimization relies on a classifier which may result in a bad classification accuracy due to the rough selection of the learning parameters. For this reason, this dissertation explores the ability of GA to search global optimum and to improve the power of the applied ML algorithms.

Unlike the previous works, this study aims to implement an effective substitute to the classical machine learning algorithms, exploring a hybrid solution combining ML, DL, and EA-based methodologies. Among the various research problems in this field, this dissertation will study the inability of single classifier methods to predict phishing websites, as referred to by Alsariera, Adeyemo, Balogun, & Alazzawi (2020). A single-classifier approach mainly produces models that are comparatively low accuracy. Moreover, it analyses the ability of genetic algorithms to compare classical feature selection algorithms, such as mutual information (MI) and recursive feature elimination, and, finally, evaluates the model's accuracy improvement with the parameter search of genetic algorithms. This dissertation presents a predictive ability comparison between different models, such as random forest (RF), extreme gradient boost (XGBoost), K-nearest neighbors (KNN), Naïve Bayes (NB), and neural networks (NN). We used a data set provided by Vrbančič et al. (2020) to evaluate the proposed solution. Besides, the proposed approach still applies to various classification problems.

This dissertation has the following organization: succeeding the introduction, section two describes the essential theoretical background which supports this dissertation, presenting concepts of ML and EA. Next, section three reviews previous and related work, whereas section four presents the methodology used in this study. Section five describes the experimental research, including the experimental settings, results, and discussion. Section 6 presents the conclusions. After summarizing the findings and drawing some conclusions, the research's limitations and recommendations will be reviewed and proposed in section seven.

2. THEORETICAL BACKGROUND

This section aims to provide the necessary theoretical knowledge to support this research and obtain a broad understanding of the scientific area in which it is inserted.

This section introduces the main concepts of machine learning and evolutionary computation, emphasizing genetic algorithms and their applications, such as feature selection and parameter optimization.

2.1. MACHINE LEARNING

Machine learning is the ability of a system to collect and process information through observations in order to learn and extend itself by acquiring new knowledge rather than being programmed with it. ML algorithms are utilized to collect insights into the data under a certain study, build a model with the phenomena abstraction, and predict future values using the generated model. (Woolf, 2009).

Over the past ten years, ML has been accountable for the surge of several technologies in various domains of study. It is possible to classify machine learning systems considering the required amount and type of supervision they receive during training, into two different concepts:

1. **Supervised learning** – Type of ML in which an algorithm learns to make predictions or decisions based on labeled examples. Labeled examples refer to a dataset in which the desired output or outcome is already known for each input or feature. Common tasks in this field are classification and regression.

2. **Unsupervised learning** – In unsupervised learning, the algorithm learns patterns and relationships in data without any explicit supervision or labeled examples. It is used when there are no predefined labels or outputs to predict. This class of algorithms searches for patterns, clusters, and other relationships in the data to uncover meaningful insights.

Evaluation metrics are required to evaluate the performance of a predictive model. There are different metrics to measure the model performance, and regarding this research, four are selected for their relevance. One of the most used evaluation metrics is accuracy - equation 2.1 - which is the ratio of the correct predictions to the total predictions. This metric is misleading when the class sizes are substantially different, therefore, it is essential to use precision - equation 2.2 – which represents the percentage of positive data points accurately identified, among the positive. The recall - equation 2.3 - is the percentage of positive samples predicted by the model out of all true positive samples. Finally, F1-score - equation 2.4 - which calculates the combination of precision and recall.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

In addition, the K[1]-Fold cross-validation technique is applied to measure the performance of the machine learning algorithms. This method is commonly used in small data sets, where the original data samples are divided into K subsets with a resampling procedure. During the validation process, one of the subsets is used, and the others are applied to the training process. Figure 1 illustrates how the available data is split during the K-fold cross-validation method when $K = 5$.

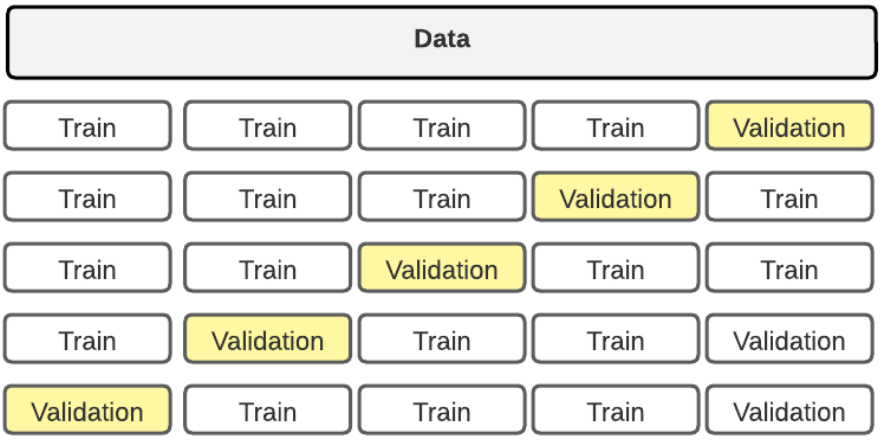


Figure 1- Example of a 5-fold cross validation approach.
Source: own illustration

2.1.1. Evolutionary Computation and Genetic Algorithms (GAs)

The term evolutionary computation emerged in 1991 with the purpose to represent researchers that have been studying different methods of simulating evolution. The subclasses of EA, which include genetic algorithms, evolution strategies, and evolutionary programming, have a fundamental common characteristic: they all imply reproduction, random variation, competition, and selection of contending individuals in a population (Baeck, B Fogel, & Michalewicz, 2000).

Genetic algorithms (GAs) are defined as EA class by John Holland (Baeck et al., 2000) and were introduced in the early 1970s (Tsoukalas & Uhrig, 1997). According to Wirsansky (2020) GAs are a "family of search algorithms inspired by the principles of evolution in nature". The algorithms reproduce the process of natural selection and reproduction and are able to produce accurate solutions for several problems, including search, optimization, and learning.

[1] $K \in (0, number\ of\ samples)$

The GA is a probabilistic search algorithm that relies on the process of natural selection and natural genetics. The algorithm initializes with a set of solutions, known as population, and a chromosome that represents a solution. The genetic algorithm follows a series of steps, shown in the following flowchart in Figure 2. In the first step, an initial population is created with the size preserved during each generation (Kumar et al., 2010).

According to Ali & Ahmed (2019), the objective of the fitness objective function is to evaluate chromosomes in the population, and then the next generation of chromosomes is selected according to the fitness value. During this process, a set of chromosomes are selected and then mate randomly, producing offspring. Offspring are produced by exchanging genes between parents until the crossover point is achieved. When producing offspring, three critical processes (also known as genetic operators) are conducted:

1. The first phase consists of the application of the selection to the individuals generated in the initialization phase without any operators. At the end of this phase, a pair of chromosomes is selected for reproduction.
2. The crossover process is a process of switching the genes across the two reproducing individuals – a crossover point is randomly chosen within the genes for each pair of parents.
3. In mutation, a particular new offspring is formed, and the genes in the chromosome are altered and replaced with randomly generated values. Mutation occurs to ensure a variety of traits within a population.

The chromosomes with higher fitness scores have a high probability of being chosen, therefore the new generation of chromosomes can have higher average fitness scores than the previous The process of evolution repeats until the conditions are fulfilled. (Kumar et al., 2010).

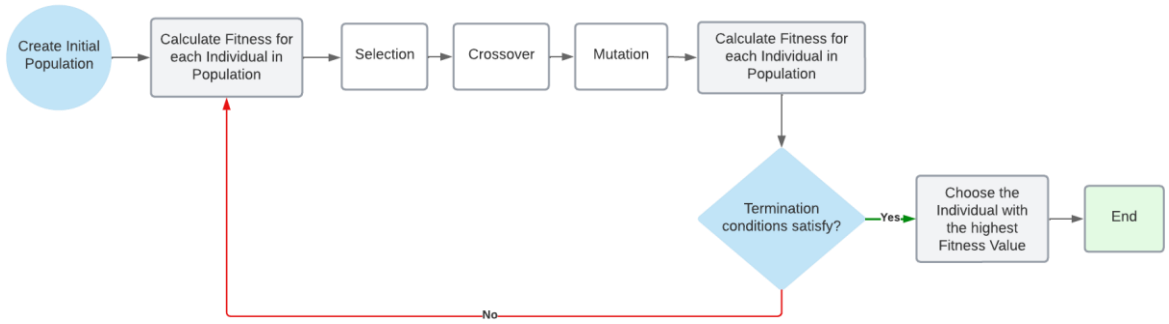


Figure 2 – Flow of a genetic algorithm.

Source: Adapted from ('Reinforcement Learning vs Genetic Algorithm — AI for Simulations | by Neelarghya | XRPractices | Medium', n.d.)

There are several advantages of GA:

- GA have the ability to be stuck in the local optimum;
- Able to solve high-complexity tasks with multiple variables and a large solution space;
- Handles multiple points in the search space concurrently, instead of an unique point, to tackle a wide parameters space;
- Parallelization possibility due to the multiple produced solutions; and
- Capable of handling multi-objective optimization.

2.1.1.1. GA in feature selection

Feature selection methods provide information regarding features that have no impact on the learning process and require meaningful ones to ensure learning performance (Banerjee, 1985). The heuristic search algorithm referred to in section 2.1.1, known as the GA algorithm, can be used as a feature selection method to determine the most effective subset of features. Following the training dataset preparation, the GA method is employed to select the most impactful features that can improve the evaluation metrics.

Figure 3 describes the feature selection with GA. The search space consists of all possible feature subsets. An individual in a population represents a feature subset corresponding to a set of genes, also known as the total number of features. Each gene contains a binary value that denotes whether a feature is selected or not (Ali & Ahmed, 2019). After setting parameters, the fitness of the selected chromosomes is ranked, and the best of these becomes the chosen one.

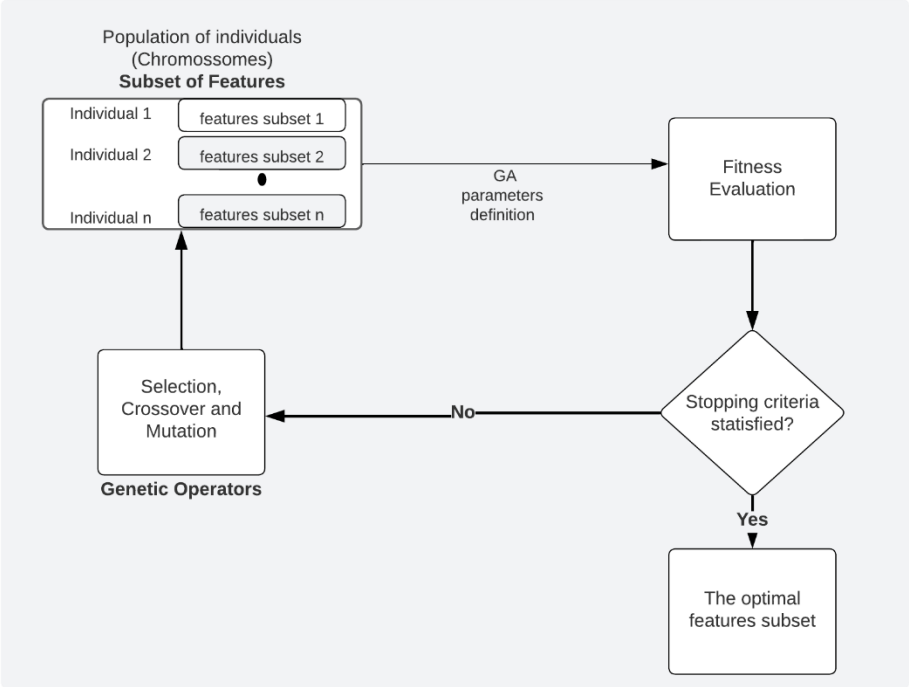


Figure 3 – Feature selection with GAs methodology.

Source: Own illustration

The evaluation metrics (represented as stopping criteria) obtained in each iteration (generation) are the following:

- The number of generations;
- The number of hyperparameters fitted in each generation;
- Average fitness score;
- The standard deviation of accuracy;
- The best score achieved in each generation; and
- The lowest score achieved in each generation.

Once the stopping criteria are not satisfied, the genetic operators are computed, and the process continues. Finally, GA will finish the search and return the optimal attributes when the requests are met.

2.1.1.2. GA in hyperparameter optimization

Starting from the idea that evolution is an optimization process, it is evident that evolution is meant to be described in terms of an algorithm that can be used to solve optimization problems. Genetic algorithms follow informative criteria to make a choice. They follow a sequential process trying to find a better set of hyperparameters by the past decisions it made, as an iterative process, making progress in each iteration and modifying the number of sets to attempt to create new ones.

A hyperparameter is a parameter not inherent to the model and is defined before the training process starts. The chosen parameters are tunable and affect the model performance. To tune them, there are specific methods, such as genetic algorithms, to determine the model's best set.

It is essential to understand a hyperparameter tuner's architecture and framework. Figure 4 shows the architecture of the GA-enhanced tuner. The tuner receives a training set from the user and returns the best fit of configuration(s) for the specified data set. The logical architecture is initialized with a set of the randomly generated set of framework configurations, which are received by the Predictive Process Monitoring Framework to produce a set of predictions for each configuration. Each framework configuration is encoded by the Configuration Tracker and the predictions are generated for the validation set. Regarding configuration metrics, the prediction evaluations according to a group of fitness functions are provided by an Evaluator. While this process occurs, Configuration Tracker collects the configurations and sends them to the Genetic Algorithm Module (di Francescomarino et al., 2018).

An evaluation function will be necessary to perform optimization with GA and determine the proximity of a potential solution, as it defines the solution fitness. The fitness function can be discrete, multimodal, etc., without mathematical restrictions. The main criteria to classify the optimization algorithms are continuous/discrete, constrained/unconstrained, and sequential/parallel (Sivanandam & Deepa, 2010).

In this sense, the GA module executes the following steps to parametrize the hyperparameters:

1. Random sample population generation, which is different sets of hyperparameters (described in Figure 4 as Configuration Metrics). Each framework configuration is considered as an individual of the population (chromosome). It is represented as a vector where each element stands for a configuration parameter;
2. Selection of configurations - the module selects the best ones, and these are used by genetic operators to create the next generation of framework configurations. The latest population of individuals/configurations is once again sent to the configuration tracker and then individually evaluated by the Predictive Process Monitoring Framework; and
3. Reach of a threshold - The process continues until a threshold is reached in terms of the number of generations or several generations without any improvement in the fitness function.

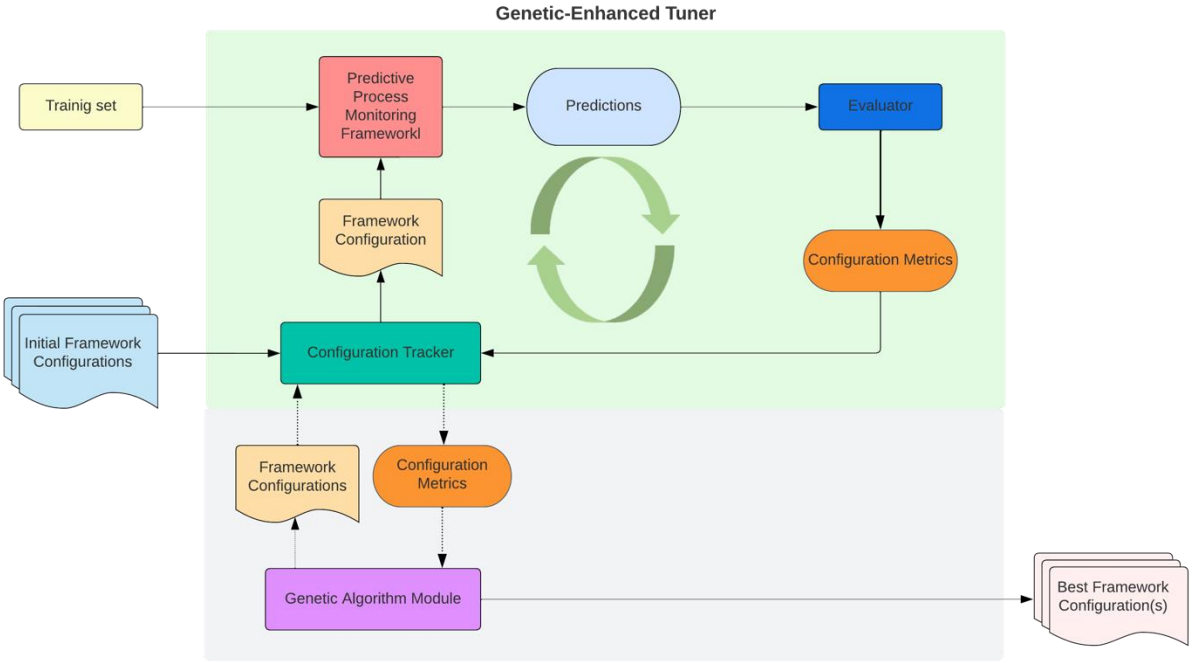


Figure 4 - Internal architecture of the Genetic-Enhanced Tuner.
Source: Adapted from (di Francescomarino et al., 2018)

"Once the genetic algorithm execution is terminated, one or more framework configurations are returned to the user as output, based on whether a single or a multi-objective approach to the optimization problem is taken. These configurations can be used to tune the Predictive Process Monitoring Framework and run it on a testing set." (di Francescomarino et al., 2018)

3. LITERATURE REVIEW

Artificial Intelligence (AI) methods are suggested to handle phishing attacks effectively, with high accuracy and low false positive rates (L. Tang & Mahmoud, 2021). The key to success is to develop an automatic anti-phishing classification system provided by website URLs. Firstly, the URLs are analyzed to perform the feature selection from phishing websites. Then, a dataset is built using extracted features along with their labels. Finally, a classification with a supervised machine learning (ML) or deep learning (DL) algorithm is used to develop a model for phishing detection.

Basit et al. (2021) proposed dividing AI phishing attack detection techniques into four categories**:** DL, ML, Scenario-based, and Hybrid learning (HL), stating that ML and DL procedures are effective strategies for detecting malicious URLs. The authors proposed that phishing website classification using DL should outperform traditional ML algorithms. Although, HL techniques that rely on combining more than one ML or DL technique to achieve good performances is another efficient way to detect phishing attacks as it occasionally has higher accuracy than ML algorithms.

Inspired by the promising ML and DL learning techniques to detect phishing websites, Almousa, Zhang, Sarrafzadeh, and Anwar (2022) proposed to research the gap in the study of features, deep learning algorithms, and hyperparameter optimization techniques for phishing website detection. The authors stated that implementing a robust detection model is crucial concerning reproducibility in other datasets to understand critical features and improve efficiency.

The introduction of genetic algorithms (GA) in URL-based phishing detection led to a variety of research in this field, with diverse applications in AI. The genetic algorithm is a probabilistic solving optimization problem that is focused on an effective algorithm to find a global optimum solution for many types of problems. According to Kumar et al. (2010), this algorithm is extremely applicable to different artificial intelligence approaches and shall concentrate on developing hybrid methods. One of these is feature selection, which GA has been known to be a highly adaptative and efficient method, as reported by Iuga, Nurse, and Erola (2016), since the users or writers can change the functional configuration of GA to improve their results further. The authors Ghatasheh, Altaharwa, & Aldebei (2022) proposed a prediction rate improvement divided into two main parts: GA features selection, which resulted in an enhanced performance compared to principal components analysis, and GA hyperparameter optimization to improve the performance of an ML algorithm.

Feature engineering is also an significant issue in phishing website detection solutions, as detection accuracy critically depends on prior knowledge of features. Suleman and Awan (2019) implemented a hybrid algorithm using ML models with GAs only for feature selection and concluded that GAs enhanced the classification of machine learning algorithms. The authors proposed using an Iterative Dichotomiser-3 (ID3) classifier along with a selection of features based on GAs, resulting in a detection accuracy of 95%.

These two concepts are addressed by the authors Ali and Ahmed (2019), who proposed an efficient HL model using deep neural networks (DNNs) with evolutionary algorithm-based for feature selection and weighting methods. The study suggested using genetic algorithms to enhance the accuracy of phishing website prediction since the conventional feature selection methods cannot identify the most appropriate website features for all datasets.

The authors proposed two different approaches with GAs: the GA-based feature selection - in which the search space of a feature selection problem is formed by all subsets of features, and the GA-based feature weighting - in which each gene in the chromosome can be assigned a weight, typically between zero and one. The results of this study showed that the suggested hybrid design for DNN-based phishing website prediction with GA-based feature selection and weighting improved the classification performance by using fewer features. Unlike authors Ali and Ahmed (2019), this study proposed a classification algorithm based only on URL attributes. Unlike the authors Ali and Ahmed (2019), this study proposed a classification algorithm based on merely URL attributes.

Aljofey, Jiang, Qu, Huang, and Niyigena (2020) proposed an effective convolutional neural network (CNN) model for phishing detection only based on URLs. In their approach, it was extracted character-level features from the original URLs, collected from phishing and benign websites. The study obtained an accuracy of 95.02% on their dataset with 318642 instances.

Bu et al. (2022) discussed the existing degradation in the deep learning approach in recall according to the nature of a phishing attack that is immediately discarded after being reported. The authors and Ali and Ahmed (2019) demonstrated a deep-learning-based URL classifier with a genetic algorithm to search for the optimal feature set that minimizes the false negatives. They proposed a genetic algorithm-embedded convolutional recurrent network, resulting in a high-accuracy phishing website detection model.

Table 1 presents a summary of the literature review on URL-based website phishing detection. Also, the table shows the more relevant studies and the main relevant evaluation metrics used in this dissertation.

Table 1 – Summary of URL-based Website Phishing Detection

Authors	**Model**	**Feature Selection Techniques**	**Dataset**	**Accuracy (%)**	**Recall (%)**	**Precision (%)**
(Chawla, 2022)	RF, DT, LR[2], KNN, ANN[3], RF, SVM	Not Provided	UCI[1]	97.37, 97.01, 92.76, 95.29, 96.38, 97.37, 95.55	99.04, 97.76, 94.08, 96.16, 97.6, 99.04, 97.12	96.41, 96.98, 93.18, 95.54, 96.06, 96.41, 95.14
(Suleman & Awan, 2019)	NB, ID3, KNN, DT, RF	GA	UCI	95	Not Provided	Not Provided

[2] Logistic Regression (LR)
[3] Artificial Neural Network (ANN)

Table 1 – Summary of URL-based Website Phishing Detection (cont.)

Authors	Model	Feature Selection Techniques	Dataset	Accuracy (%)	Recall (%)	Precision (%)
(Ali & Ahmed, 2019)	DNN	GA for feature selection and weighting	UCI	91.13	90.79	Not Provided
(Bu et al., 2022)	CNN	GA	ISCX-URL-2016[2], PhishStorm[3], PhishTank[4]	96.85 95.05 94.83	95.10 93.32 90.81	Not Provided
(Aljofey, Jiang, Qu, Huang, & Niyigena, 2020b)	MNB, LR, NB, RF, XGB, DNN, CNN	Not Provided	Yandex[5], Common-craw[6], Phishtank, Alexa[7]	87.44, 91.83, 80.43, 93.62, 92.43, 95.24, 95.41	70.16 88.97, 89.63, 84.11, 90.66, 93.93, 94.31	Not Provided
(Gandotra & Gupta, 2021)	RF NB DT KNN SVM Adaboost	Not Provided	GSB database[8]	99.5 85.7 90 93.3 95.9 98.5	Not Provided	99.4 84.9 89.2 92.4 94.7 98.4
(Lakshmi, Mittapalli, Santhaiah, & Reddy, 2021)	DNN	MI	UCI	Not Provided	Not Provided	96.25

[1] https://archive.ics.uci.edu/ml/index.php
[2] https://www.unb.ca/cic/datasets/url-2016.html
[3] https://research.aalto.fi/en/datasets/phishstorm-phishing-legitimate-url-dataset
[4] https://phishtank.org/
[5] https://yandex.com/
[6] https://commoncrawl.org/
[7] https://www.alexa.com/
[8] https://www.gsb.stanford.edu/library/research-resources/databases

The main findings presented in Table 1 are:

- The ML and DL prediction models most applied are RF, KNN, DNN, XGB, and NB;
- Several authors applied adaptations of GA feature selection methods, like GA, in feature selection and weighting. Also, mutual Information was referred to, among others;
- The evaluation, specifically in accuracy, present high values.

4. METHODOLOGY

The research methodology presents the applied methods and respective steps during the elaboration of this dissertation. The research comprises three main phases (Figure 5): the Exploration, where the Literature and Methodology Review are inserted, followed by the Modeling phase, composed of the experimental study planning, finalizing with the Conclusion and Discussion.

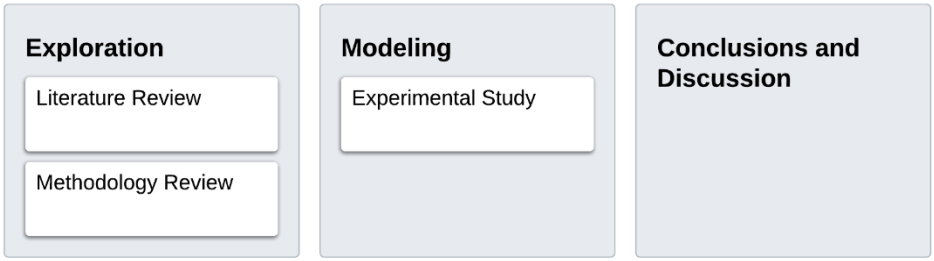


Figure 5 - Methodology main phases.
Source: Own illustration

The exploration phase reviewed the dissertation's fundamental concepts, demonstrating the necessary knowledge on several topics, such as research on AI phishing detection and promising methodologies URLs featured-based and hybrid learning methods. This consists of applying and developing ML, DL and GA algorithms in this field and introducing genetic algorithms in feature selection and parameter hyper tuning.

CRISP-DM (Cross Industry Standard Process for Data Mining) supports the conceptual model used in the second stage. The CRISP-DM methodology contains the phases of a project, their respective tasks, and outputs. A data mining project's life cycle consists of six phases shown in Figure 6.

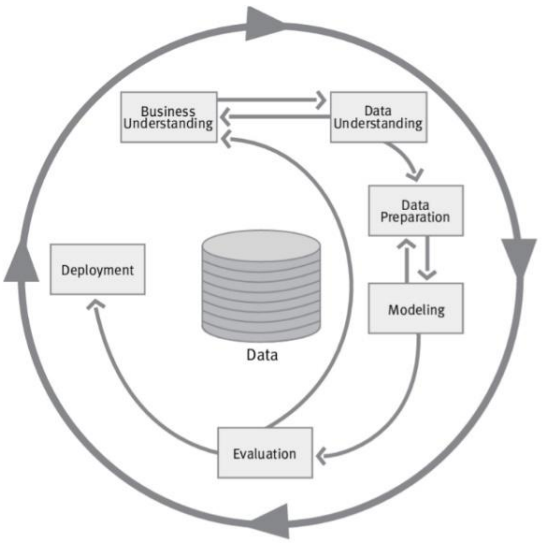


Figure 6 - CRISP-DM methodology.
Source: ('Data Mining Process', n.d.)

Figure 6 describes the following phases:

- The initial phase, **Business Understanding,** focuses on understanding the project requirements and objectives. In the research, this process is defined based on the outstanding necessity of an intelligent websites phishing detection model;
- In **Data Understanding**, initial data collection proceeds with activities to get familiar with the data, identify data quality issues, and discover insights into the data. In this stage, the data preprocessing is approached in Sections 5.1 and 5.2, covering missing values analysis and data visualization with adequate methods;
- The **Data Preparation** phase covers all the transformations and cleaning on the dataset. The aim of this step is mainly, the construction of the final dataset to be used in the models. Therefore, this research conducted the missing values treatment, feature encoding and data partition and sampling;
- Several models are selected and applied in the Modelling Phase, and their parameters are calibrated to optimal values. Section 5.3 addresses this stage of the feature selection process, where three methods are applied to the training dataset. The chosen feature subsets are trained with classification models, and the parameters are tuned, (described in Section 5.4);
- **Evaluation** proceeds before deployment, which is the phase where the models are evaluated with appropriate metrics. Before the deployment, a review stage is important to verify if the business requirements and objectives are being fulfilled. This stage is mentioned in Section 5.5 of this dissertation, where the adequate evaluation metrics are presented; and
- **The deployment** comes in the final stage, where the knowledge and model results must be presented and organized. This process usually involves plan monitoring and maintenance to ensure the model's quality and applicability.

5. EXPERIMENTAL STUDY

5.1. DATASET

The chosen dataset was provided originally from a published paper by the authors Vrbančič et al. (2020). This paper presents a dataset comprising 88647 websites labeled as legitimate or phishing and allows the researchers to train their classification models, build phishing detection systems, and mining association rules (Vrbančič et al., 2020). The dataset was provided with 111 attributes, about 65,4% are labeled as legitimate website URLs, and the rest are confirmed phishing URLs. This specific subset was extracted from PhishTank[4] and Alexa ranking websites[5]. The independent variables are based on the URL properties, URL resolving metrics, and external services, such as attributes based on the whole URL, domain, directory properties, file, parameter, resolving data, and external metrics. The dependent variable - target - characterizes if the URL is legitimate or illegitimate. The target class 0 denotes legitimate websites, while the target class 1 denotes phishing websites.

The high number of features makes it a high-dimensional dataset, so ordinary methods can't be used to plot the dataset. T-Distributed Stochastic Neighbor Embedding (T-SNE) is used for dimensional reduction by reducing 45 attributes into two dimensions for visualization and exploratory data analysis of the dataset (Figure 7).

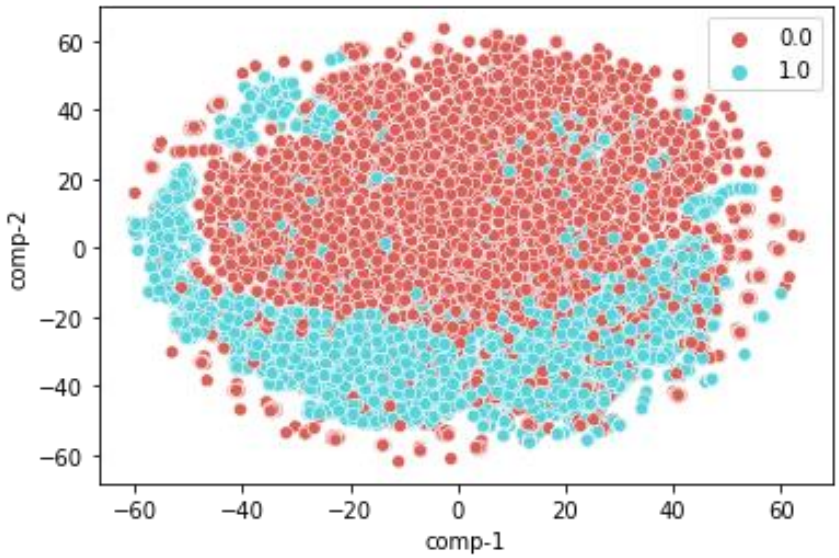


Figure 7 - Scatter plot of Component 1 ("comp-1") and Component 2 ("comp-2") from the result obtained from dimensionality reduction of the original dataset. The color scheme is based on the output variable where 1 indicates phishing website and 0 legitimate website.

Source: own illustration

Figure 7 shows that several sample points are overlapping, although most points can be separated using a non-linear classification model.

[4] https://phishtank.org/

[5] https://www.alexa.com

5.2. DATA PREPROCESSING

The preprocessing section is mainly covered by three successive processes applied to the initial phishing websites dataset to assemble the entries into numerical features that can be the input for machine learning algorithms to provide proper inferences and decisions. These processes are dataset import, reshaping, and preparation of training data set. The preprocessing stage of the substantial website's features is an essential step that significantly impacts the performance of phishing website detection techniques.

During the preparation stage of the training dataset, the extracted features are transformed into numerical or categorical attributes that can be utilized in classifiers like DNN. All the accessed experiments were done with a total of 45 variables due to eliminating missing data assigned with -1.

5.2.1. Data Partition and Sampling

Once the training data set is prepared correctly, the splitting ratio used in this research is 70% for training and 30% for testing. Also, due to computational limitations, a sample was created based on the same proportions of phishing and legitimate URLs. This was done with the Scikit Learn[6] library in Python. From here on, until the evaluation, the Scikit Learn library was used in the implementation.

5.3. FEATURE SELECTION

In this section, for the classification problem, the implemented feature selection algorithms aim to select a minimally sized subset of highly discriminant features. Three methods are applied: mutual information, recursive feature elimination, and genetic algorithms.

5.3.1. Mutual Information

Mutual Information (MI) is an impurity-based measure for classification problems and works on the entropy of the variables. Furthermore, "MI measures the amount of information one can obtain from one random variable given another" (Witten, Frank, & Hall, 2011). The mutual information between two random variables X and Y can be formally stated as follows:

$$I(X;Y) = H(X) - H(H|Y) \qquad (5.3.1.1)$$

Where $I(X;Y)$ is the mutual information for X and Y, $H(X)$ is the entropy for X, and $H(X|Y)$ is the conditional entropy for X given Y.

In information gain, a feature is relevant if it has a high information gain. Features are selected univariately; therefore, information gain cannot handle redundant features (J. Tang, Alelyani, & Liu, 2014). Mutual Information is a powerful method that may prove useful for both categorical and numerical data, e.g., it is agnostic to the data types and can also detect non-linear dependencies among variables. It is a measure of "how much information (in terms of entropy) two random variables share. ("How to Choose a Feature Selection Method For Machine Learning", n.d.)

[6] https://scikit-learn.org/stable/

"MI between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency" (scikit-learn developers, n.d.).

According to the MI method results (Appendix 1) there are 30 relevant variables where the MI is superior to 0. It is evident from the analysis of Figure 8 that the feature "qty_slash_url", followed by "length_url" with the highest score, is the most relevant feature. The variables with the lowest worth are "qty_plus_domain", "qty_comma_domain", "qty_dollar_url", "qty_tilde_domain", "qty_asterisk_url", "qty_and_domain", "qty_space_url", "qty_equal_domain", "qty_underline_domain".

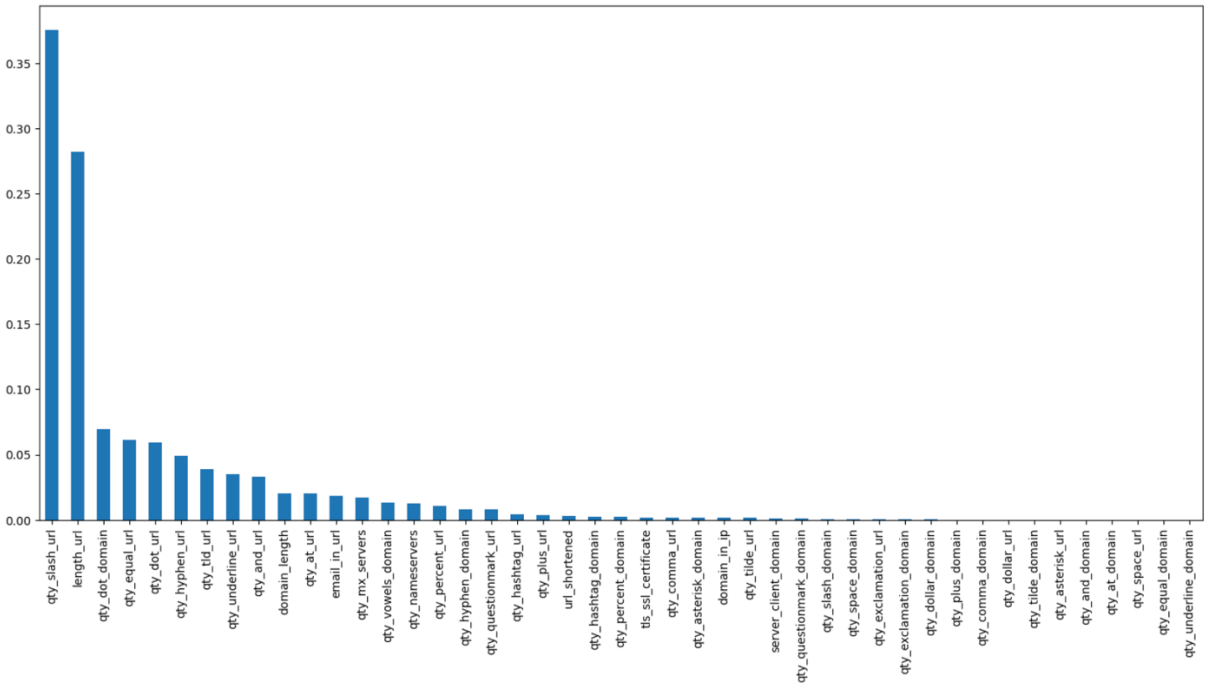


Figure 8 - Variable Importance with Mutual Information.
(Source: Python Output, 2022)

5.3.2. Recursive Feature Elimination

Guyon, Weston, Barnhill and Vapnik (2002) described a backward selection algorithm called recursive feature elimination (RFE) that avoids refitting many models at each step of the search (Kuhn & Johnson, 2013). RFE is a wrapper-type feature selection algorithm. This means that a different machine learning algorithm (e.g., the random forest importance criterion) is set in the core of the method, which is wrapped by RFE and used to help select features. RFE removes the weakest feature (or features) until a specified number of features is reached. The model's coefficient attributes rank featurs, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model ("Recursive Feature Elimination — Yellowbrick v1.5 documentation", n.d.)

A recursive feature elimination example with automatic tuning of the number of features selected with cross-validation was implemented. To avoid the overfitting problem, it was applied 3-fold cross-fold validation. The method was trained and evaluated with three classifiers - Decision Tree, Random Forest, and Gradient Boosting – presented in Appendix 2. According to the model evaluation, the selected model was RF due to the high accuracy result, and the selected features were: "qty_dot_url", "qty_hyphen_url",

"qty_underline_url", "qty_slash_url", "qty_equal_url", "length_url", "qty_dot_domain", "qty_vowels_domain", "domain_length", "qty_nameservers" and "qty_mx_servers".

5.3.3. Genetic Selection

Figure 9 describes the process steps of the GA feature selection implementation: First, the collection phase of the data set of phishing and legitimate URLs, followed by the preprocessing phase mentioned in Section 5.2. The following phase describes the GA feature selection complete process. To compute the GA, it is essential to define several parameters (presented in Table), such as population size, the number of generations, the crossover and mutation probability, and the estimator. Based on the GA configuration in Table and a length of 45 in each chromosome, since the experimental data set has a total of 45 features, the GA was implemented. The maximum number of generations was set to 20 to avoid GA being trapped in the local optimum. Furthermore, the Support Vector Machine (SVM) model was chosen as the estimator - the algorithm fits the model and evaluates the fitness mean with the results from cross-validation evaluation with $k = 3$.

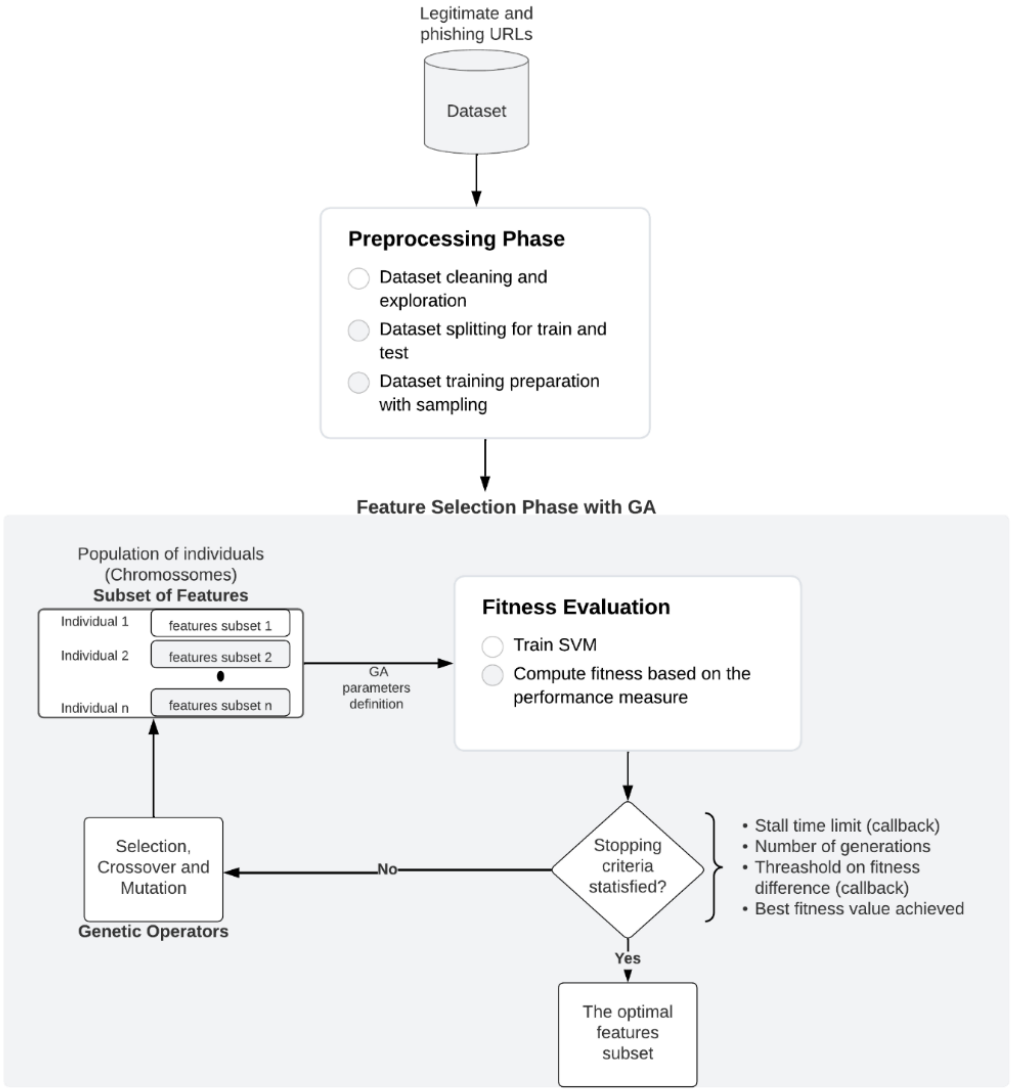


Figure 9 - Methodology of the proposed approach with feature selection GA-based.
(Source: own illustration)

Table 2 - Parameters settings of GA used with GA-based feature selection.

Parameter	Value
Estimator	Support vector machine (SVM)
Population size	30
Number of generations	20
Crossover probability	0.9
Mutation probability	0.05
Algorithm	eaSimple
Evaluation	Cross validation with k=3

According to Figure 9, due to computational limitations, besides the number of generations and optimum fitness value achievement, a set of conditions were added to the stopping criteria, such as a stall time limit of 200 seconds and a delta threshold of 0.001 on the fitness metric, (if the difference on the latest fitness upgrade was inferior to 0.001, the algorithm automatically stops). Finally, the GA using the RF estimator terminates at generation 15 with a selection of 21 features and a fitness mean of 92.7%. The selected features were: “qty_dot_url”, “qty_hyphen_url”, “qty_underline_url”, “qty_slash_url”, “qty_equal_url”, qty_at_url”, “qty_and_url”, “qty_hashtag_url”, “qty_percent_url”, “length_url”, “qty_dot_domain”, “qty_hyphen_domain”, “qty_tilde_domain”, “qty_plus_domain”, “qty_vowels_domain”, “domain_length”, “domain_in_ip”, “server_client_domain”, “qty_nameservers”, “qty_mx_servers” and “tls_ssl_certificate”.

The results based on this selection were evaluated and discussed in the next section. The use of GA as a feature selection algorithm was done using the sklearn-genetic-opt library in Python[7] with the function GAFeatureSelectionCV. This library uses evolutionary algorithms to fine-tune scikit-learn machine-learning algorithms and perform feature selection.

5.4. Modeling Results and Discussion

To describe this section, Figure 10 presents the main steps of modeling and evaluation. To evaluate the performance of the URL features and compare the GA feature selection with MI and RFECV, different classification models were applied, such as Random Forest, XGBoost, Naïve Bayes, KNN, and NN. The main goal of comparing different models with different feature subsets is to expose the best classifiers suitable for the URL group features and compare the results based on the feature selection method applied. The prediction approaches are chosen based on the previous feature selection methodologies and, consequently, improved with the GA hyperparameter algorithm. Once the classification models are trained by considering GA-based feature selection, RFECV and MI, the results are evaluated with unseen testing data.

[7] (‘sklearn-genetic-opt — sklearn genetic opt 0.9.0 documentation’, n.d.)

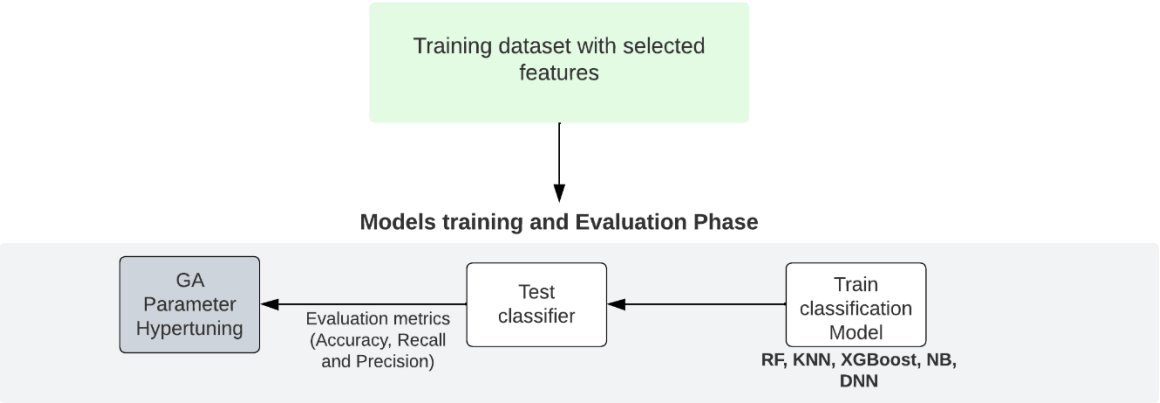


Figure 10 - Methodology process in the Modeling phase.
(Source: own illustration)

Regarding the performance evaluation of the proposed URL prediction methods, it is suitable to use evaluation metrics such as recall (equation 2.3), precision (equation 2.2), accuracy (equation 2.1), and f1-score (equation 2.4).

The evaluation of the classification algorithms' performance based on MI, RFE, and GA feature selection methods was computed. In order to precisely evaluate the proposed methods, the algorithms were trained and assessed with 10-cross validation. The comparison results are shown in Appendix 3. The experiment results without the hyperparameter tuning (Appendix 3) show that RF, XGBoost, and KNN have higher accuracy, precision, recall, and F1-score with GA feature subsets. Though, as can be seen in Figure 11, **RF with GA-based features** achieved the highest accuracy (94.27%) among the classifiers.

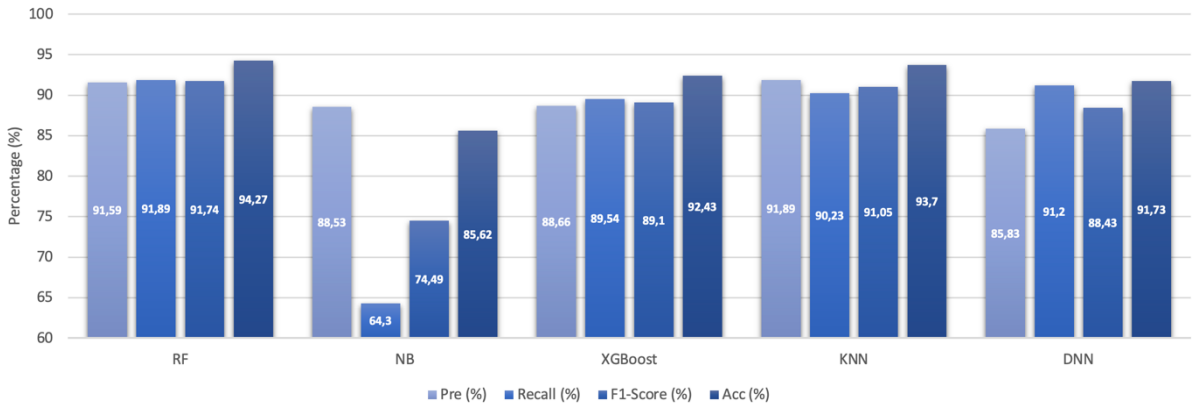


Figure 11 - Comparison of KNN, NB, DNN, RF, and XGBoost with GA feature selection before GA optimization.
Source: Own Illustration

To enhance the performance of the best classification models (RF, KNN, and XGBoost), the EA hyperparameter optimization was obtained. Several EA optimization experiments were done to maximize the fitness value. The use of GA as an optimization algorithm was done using the sklearn-

genetic-opt library in Python[8] with the function GASearchCV. This library is the object that allows the implementation of the fitting process using evolutionary algorithms.

To implement GASearchCV, several parameters need to be specified. The parameters used in this research are the cross-validation evaluation, number of generations, scoring, population size, mutation and crossover probability, and the algorithm. Table 5 presentes the parameters used in the random forest classifier. Some of the experiments were computed using a mutation and crossover adapter. Adapters control the form and the speed from which the parameter goes from an initial value to a final threshold. In this research, the exponential adapter ranges between 0.05 and 0.8 for the mutation probability and between 0.2 and 0.9 for crossover. Also, experiments reveal that the number of iterations (i.e., population size and generations) reaches a relatively local maximum early. Consequently, the number of generations is lowered to 25 considering the effect on larger initial populations and the number of features.

Table 3 - Parameters settings of GASearchCV for parameter hyper tuning in random forest.

Parameter	Value
Cross validation	3
Generations	25
Population size	20
Scoring	accuracy
Crossover probability	ExponentialAdapter(initial_value = 0.2, end_value = 0.9, adaptative_rate = 0.1)
Mutation probability	ExponentialAdapter(initial_value = 0.05, end_value = 0.8, adaptative_rate=0.1)
Algorithm	eaSimple

Regarding evaluation, the experiences were conducted with 3-cross validation.

Each classifier has a different hyperparameter to optimize. In RF, the chromosome of the GA is the estimators' number, maximum depth, minimum samples split, minimum samples leaf, and maximum features. Table 6 presents the parameter settings resulting from GA optimization and used for RF training.

Table 4 - Parameters settings of random forest classifier resultant from EA optimization.

Parameter	Value
Estimators Number	1691
Maximum Depth	608

[8] ('sklearn-genetic-opt — sklearn genetic opt 0.9.0 documentation', n.d.)

Table 4 - Parameters settings of random forest classifier resultant from EA optimization. (cont.)

Minimum samples split	**5**
Minimum samples leaf	1
Maximum features	Log2

Once the predictions with RF, KNN, DNN, and XGBoosting with hyperparameter tuning are performed, their results are assessed and compared. According to Appendix 5, random forest with GA-feature selection achieved the highest accuracy (95.34%) and F1-score (92.67%). Also, among the classifiers, we can notice that the classification evaluation metrics of RF, KNN, XGBoost, and DNN were improved by applying the GA hyperparameter tuning optimization. It is clear from Figure 12 that GA-based feature selection and optimization methods contributed to enhancing the performance in terms of the classification accuracy of most classifiers utilized in the prediction of phishing websites. However, due to the significantly low performance of NB, the algorithm was not considered for parameter optimization. Regarding precision and recall, the results in Appendix 5 and Figure 12 show that most of the machine learning classifiers performed better after applying GA-based feature selection and EA optimization.

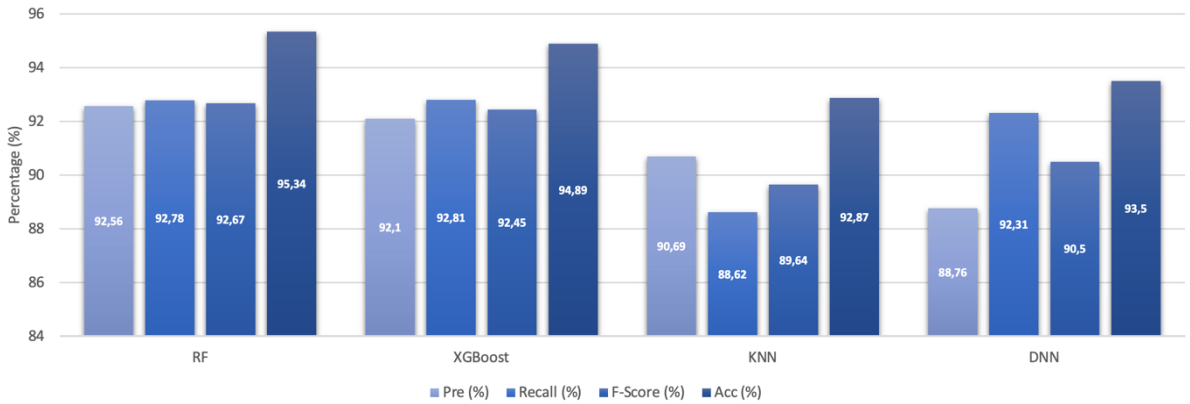


Figure 12 - Comparison of KNN, NB, DNN, RF, and XGBoost with GA feature selection after GA optimization.

Source: Own Illustration

Following the overall model performance of RF with the GA feature selection subset, the analysis of the estimated SHAP values[9] is presented in Figure 13, which depicts a summary plot of estimated SHAP values colored by feature values for all main feature effects and their interaction effects, ranked from top to bottom by their importance (Li, 2022). A high feature value corresponds to a high probability of a positive (phishing website) or negative (legitimate website) output value. According to Figure 13, the "qty_slash_url" has the highest impact on the model output, whether the target is positive or negative. For instance, we see that higher values of "qty_slash_url" have positive SHAP values (the points extending towards the right are increasingly red) and lower values of "qty_slash_url" have negative SHAP values (the points extending towards the left are increasingly blue). This indicates that URLs with a higher number of "qty_slash_url" are mostly phishing websites and vice versa. The reverse is seen

[9] **SHAP (SHapley Additive exPlanations)** is a game theoretic approach to explain the output of any machine learning model. SHAP quantifies how important each input variable is to a model for making predictions.

for “qty_dot_domain” and “qty_mx_servers”- lower quantity of dots and servers counts lead to phishing URLs.

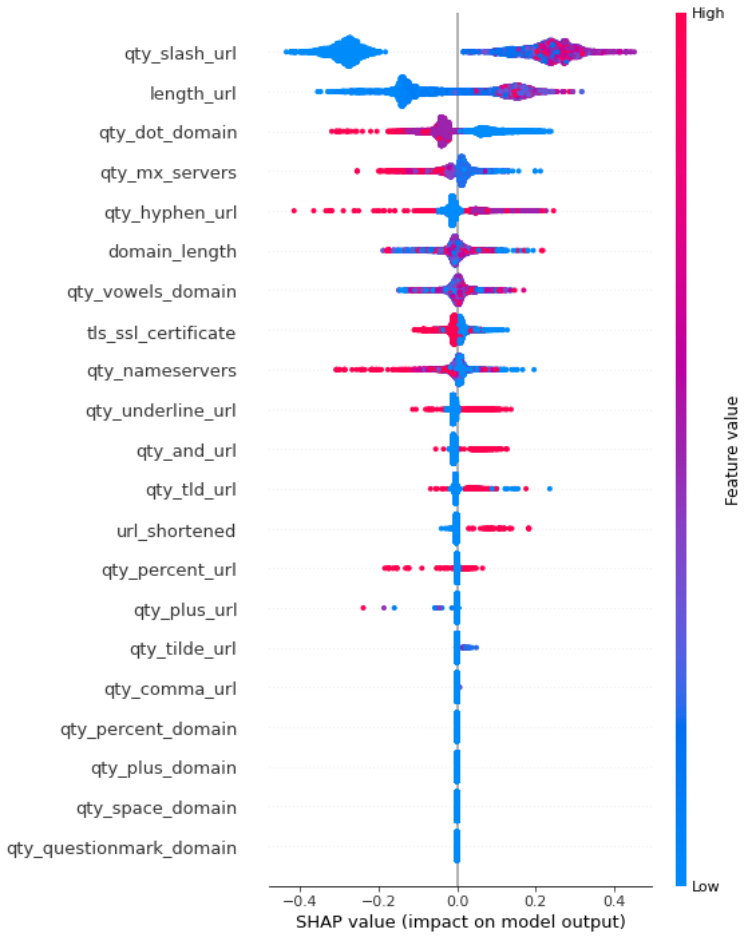


Figure 13 – Beeswarm plot, ranked by mean absolute SHAP value.
Source: Own illustation (Python Output)

6. CONCLUSIONS

This dissertation presents a hybrid solution-based model for phishing website detection through URLs. The proposed model uses a set of 21 features that are categorized into different categories, i.e., URL, domain, file, and directory properties-based features. The performance of each category of features is evaluated and compared with the integration of genetic algorithms methodologies, such as feature selection and optimization. A genetic algorithm is a probabilistic solving optimization problem modeled on a genetic evaluation process in biology and is focused on an effective algorithm to find the global optimum solutions for many types of problems (Kumar et al., 2010). This class of evolutionary algorithms has been shown to have great potential in many applications.

The proposed solution with GA feature selection and parameters optimization demonstrates that combining machine learning and GA has resulted in an enhanced URL phishing website detection solution. Regarding feature selection, GA outperformed some of the most commonly used methods in the literature, such as mutual information based on the entropy calculus and recursive feature elimination, a wrapper-type feature selection algorithm. This demonstrated that the proposed hybrid phishing website prediction approaches based on machine learning and GA feature selection enhanced the classification performance using fewer features.

Furthermore, the performance results also report a significant improvement when integration of GA parameter hyper tuning is integrated. Again, random forest is the best classifier as it reaches an accuracy of 95.34% with a recall and precision of 92.78% and 92.56%, respectively. This research has shown alternative solutions to effectively predict phishing websites using URL features to provide more confidence to online users.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

The stochastic aspects of genetic algorithms and parameter optimization are among the research limitations. Genetic algorithm-based solutions usually require a large number of initial population spaces or a large number of generations to find an outperforming solution. It takes considerable computation time to have a relatively robust URL phishing detection model using GA for feature selection and hyperparameter tuning. Due to this limitation, two stopping conditions were implemented in the GA algorithms, for running time and for a minimum improved fitness.

Furthermore, the proposed dataset from (Vrban ˇci ˇc et al., 2020) only contained pre-extracted features from the URL; hence, its scope is narrower, and the application of the dataset is limited. A proposal for future work would be the addition of an extended feature set and deep learning algorithms for improving the classification accuracy of detecting fishing websites at a large scale.

8. BIBLIOGRAPHY

Ali, W., & Ahmed, A. A. (2019). Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting. *IET Information Security*, 13(6). Retrieved from https://doi.org/10.1049/iet-ifs.2019.0006

Aljofey, A., Jiang, Q., Qu, Q., Huang, M., & Niyigena, J. P. (2020b). An effective phishing detection model based on character level convolutional neural network from URL. *Electronics (Switzerland)*, 9(9), 1–24. Retrieved 9 October 2022 from https://doi.org/10.3390/ELECTRONICS9091514

Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing Attacks: A Recent Comprehensive Study and a New Anatomy. *Frontiers in Computer Science*, 3. Retrieved from https://doi.org/10.3389/fcomp.2021.563060

Almousa, M., Zhang, T., Sarrafzadeh, A., & Anwar, M. (2022). Phishing website detection: How effective are deep learning-based models and hyperparameter optimization? Retrieved from https://doi.org/10.1002/spy2.256

Alsariera, Y. A., Adeyemo, V. E., Balogun, A. O., & Alazzawi, A. K. (2020). AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites. *IEEE Access*, 8. Retrieved from https://doi.org/10.1109/ACCESS.2020.3013699

Baeck, T., B Fogel, D., & Michalewicz, Z. (2000). *Evolutionary Computation 1 Basic Algorithms and Operators*. (M. Beavis,Ed.). Bristol and Philadelphia: Institute of Physics Publishing.

Banerjee, S. (1985). Computational methods. Retrieved from https://doi.org/10.1785/bssa0580041339

Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., & Kifayat, K. (2021). A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems*. Retrieved from https://doi.org/10.1007/s11235-020-00733-2

Bu, S.-J. ;, Kim, H.-J., Radoglou-Grammatikis, P., Sarigiannidis, P., Lagkas, T., Argyriou, V., … Kim, H.-J. (2022). Citation: Optimized URL Feature Selection Based on Genetic-Algorithm-Embedded Deep Learning for Phishing Website Detection. Optimized URL Feature Selection Based on Genetic-Algorithm-Embedded Deep Learning for Phishing Website Detection, 11, 1090. Retrieved from https://doi.org/10.3390/electronics11071090

Chawla, A. (2022). Phishing website analysis and detection using Machine Learning. *International Journal of Intelligent Systems and Applications in Engineering*, 10(1), 10–16. Retrieved 9 October 2022 from https://doi.org/10.18201/ijisae.2022.262

Data Mining Process. (n.d.). Retrieved 18 January 2023, from http://statshacker.com/blog/2018/04/14/data-mining-process/

di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., Rizzi, W., & Simonetto, L. (2018). Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems*, 74, 67–83. Retrieved from https://doi.org/https://doi.org/10.1016/j.is.2018.01.003

Gandotra, E., & Gupta, D. (2021). Improving Spoofed Website Detection Using Machine Learning. *Cybernetics and Systems*, 52(2), 169–190. Retrieved from https://doi.org/10.1080/01969722.2020.1826659

Ghatasheh, N., Altaharwa, I., & Aldebei, K. (2022). Modified Genetic Algorithm for Feature Selection and Hyper Parameter Optimization: Case of XGBoost in Spam Prediction. *IEEE Access*. Retrieved 3 October 2022 from https://doi.org/10.1109/ACCESS.2022.3196905

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning 2002 46:1*, 46(1), 389–422. Retrieved 6 November 2022 from https://doi.org/10.1023/A:1012487302797

How to Choose a Feature Selection Method For Machine Learning. (n.d.). Retrieved 6 November 2022, from https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

Iuga, C., Nurse, J. R. C., & Erola, A. (2016a). Baiting the hook: factors impacting susceptibility to phishing attacks. *Human-Centric Computing and Information Sciences*, 6(1), 8. Retrieved from https://doi.org/10.1186/s13673-016-0065-2

Iuga, C., Nurse, J. R. C., & Erola, A. (2016b). Baiting the hook: factors impacting susceptibility to phishing attacks. *Human-Centric Computing and Information Sciences*, 6(1), 8. Retrieved from https://doi.org/10.1186/s13673-016-0065-2

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. *Applied Predictive Modeling*. Retrieved from https://doi.org/10.1007/978-1-4614-6849-3

Kumar, M., Husain, Dr. M., Upreti, N., & Gupta, D. (2010). Genetic Algorithm: Review and Application. *SSRN Electronic Journal*. Retrieved 19 November 2022 from https://doi.org/10.2139/SSRN.3529843

Lakshmi, L., Mittapalli, P., Santhaiah, C., & Reddy, U. (2021). Smart Phishing Detection in Web Pages using Supervised Deep Learning Classification and Optimization Technique ADAM. *Wireless Personal Communications*, 118. Retrieved from https://doi.org/10.1007/s11277-021-08196-7

Li, Z. (2022). Extracting spatial effects from machine learning model using local interpretation method: An example of SHAP and XGBoost. *Computers, Environment and Urban Systems*, 96, 101845. Retrieved from https://doi.org/https://doi.org/10.1016/j.compenvurbsys.2022.101845

Recursive Feature Elimination — Yellowbrick v1.5 documentation. (n.d.). Retrieved 6 November 2022, from https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html

Reinforcement Learning vs Genetic Algorithm — AI for Simulations | by Neelarghya | XRPractices | Medium. (n.d.). Retrieved 19 November 2022, from

https://medium.com/xrpractices/reinforcement-learning-vs-genetic-algorithm-ai-for-simulations-f1f484969c56

scikit-learn developers. (n.d.). scikit-learn. Retrieved 6 November 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html#r37d39d7589e2-1

Sivanandam, S. N., & Deepa, S. N. (2010). *Introduction to Genetic Algorithms* (1st ed.). Springer Publishing Company, Incorporated.

sklearn-genetic-opt — sklearn genetic opt 0.9.0 documentation. (n.d.). Retrieved 12 January 2023, from https://sklearn-genetic-opt.readthedocs.io/en/stable/

Suleman, M. T., & Awan, S. M. (2019). Optimization of URL-Based Phishing Websites Detection through Genetic Algorithms. *Automatic Control and Computer Sciences*, 53(4). Retrieved from https://doi.org/10.3103/S0146411619040102

Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, 37–64. Retrieved from https://doi.org/10.1201/b17320

Tang, L., & Mahmoud, Q. H. (2021). A Survey of Machine Learning-Based Solutions for Phishing Website Detection. *Machine Learning and Knowledge Extraction*, 3(3). Retrieved from https://doi.org/10.3390/make3030034

Tsoukalas, L. H., & Uhrig, R. E. (1997). Fuzzy and neural approaches in engineering. Retrieved 19 November 2022 from http://localhost:8080/xmlui/handle/123456789/909

Vrbančič, G., Fister, I., & Podgorelec, V. (2020). Datasets for phishing websites detection. *Data in Brief*, 33. Retrieved from https://doi.org/10.1016/j.dib.2020.106438

Wirsansky, E. (2020). *Hands-On Genetic Algorithms with Python*. Packt Publishing Ltd.

Witten, I. H., Frank, E., & Hall, M. A. (2011). Chapter 7 - Data Transformations. In I. H. Witten, E. Frank, & M. A. Hall (Eds.), *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)* (pp. 305–349). Boston: Morgan Kaufmann. Retrieved from https://doi.org/https://doi.org/10.1016/B978-0-12-374856-0.00007-9

Woolf, B. P. (2009). Building Intelligent Interactive Tutors. *Building Intelligent Interactive Tutors*. Retrieved 28 December 2022 from https://doi.org/10.1016/B978-0-12-373594-2.X0001-9

Yang, P., Zhao, G., & Zeng, P. (2019). Phishing website detection based on multidimensional features driven by deep learning. *IEEE Access*, 7. Retrieved from https://doi.org/10.1109/ACCESS.2019.2892066

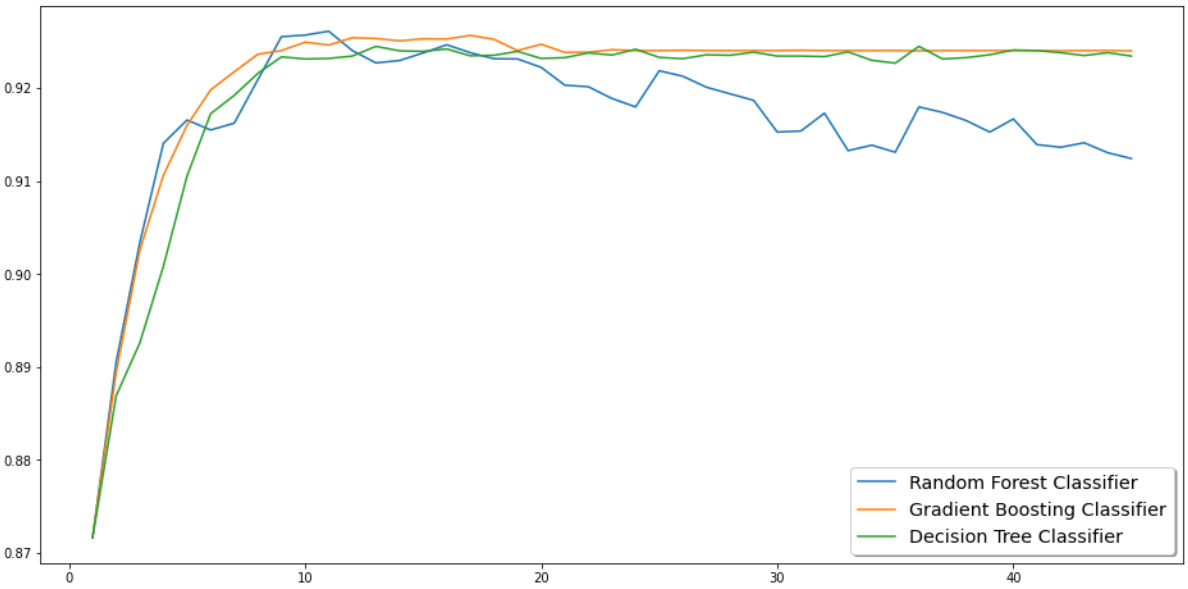
9. APPENDIX

- **Appendix 1: Python Output with Mutual Information Score**

Variable	MI Score
qty_slash_url	0.3757
length_url	0.2823
qty_dot_domain	0.0697
qty_equal_url	0.0611
qty_dot_url	0.0592
qty_hyphen_url	0.0492
qty_tld_url	0.0386
qty_underline_url	0.0350
qty_and_url	0.0329
domain_length	0.0201
qty_at_url	0.0200
email_in_url	0.0181
qty_mx_servers	0.0168
qty_vowels_domain	0.0135
qty_nameservers	0.0125
qty_percent_url	0.0104
qty_hyphen_domain	0.0080
qty_questionmark_url	0.0078
qty_hashtag_url	0.0040
qty_plus_url	0.0037
url_shortened	0.0032
qty_hashtag_domain	0.0026
qty_percent_domain	0.0020
tls_ssl_certificate	0.0019
qty_comma_url	0.0019
qty_asterisk_domain	0.0018
domain_in_ip	0.0017
qty_tilde_url	0.0017
server_client_domain	0.0013
qty_questionmark_domain	0.0013
qty_slash_domain	0.0006
qty_space_domain	0.0004
qty_exclamation_url	0.0004
qty_exclamation_domain	0.0002
qty_dollar_domain	0.0001
qty_plus_domain	0
qty_comma_domain	0

qty_dollar_url	0
qty_tilde_domain	0
qty_asterisk_url	0
qty_and_domain	0
qty_at_domain	0
qty_space_url	0
qty_equal_domain	0
qty_underline_domain	0

- **Appendix 2: Accuracy metric evolution of RFECV with RF, GB, and DT, accordingly to the number of selected features**



- **Appendix 3: Python Output with Feature Selection for MI, RFECV and GA**

Feature Selection Method	Classifier	Number of Selected Features	Top Relevant Features
MI	-	30	'qty_dot_url', 'qty_hyphen_url', 'qty_underline_url', 'qty_slash_url', 'qty_questionmark_url', 'qty_equal_url', 'qty_at_url', 'qty_and_url', 'qty_exclamation_url', 'qty_tilde_url', 'qty_asterisk_url', 'qty_dollar_url', 'qty_percent_url', 'qty_tld_url', 'length_url', 'qty_hyphen_url', 'qty_underline_url', 'qty_slash_url', 'qty_questionmark_url', 'qty_equal_url', 'qty_at_url', 'qty_and_url', 'qty_exclamation_url', 'qty_tilde_url', 'qty_asterisk_url', 'qty_dollar_url', 'qty_percent_url', 'qty_tld_url', 'length_url', 'qty_dot_domain', 'qty_hyphen_domain', 'qty_and_domain', 'qty_comma_domain', 'qty_dollar_domain', 'qty_percent_domain', 'qty_vowels_domain', 'domain_length', 'domain_in_ip', 'server_client_domain', 'email_in_url', 'qty_nameservers', 'qty_mx_servers', 'tls_ssl_certificate', 'url_shortened'
RFECV	Decision Tree	13	‘qty_dot_url’, ‘qty_hyphen_url’, ‘qty_underline_url’, ‘qty_slash_url’, ‘qty_equal_url’, ‘length_url’, ‘qty_dot_domain’, ‘qty_hyphen_domain’, ‘qty_vowels_domain’, ‘domain_length’, ‘qty_nameservers’,

			‘qty_mx_servers’, ‘tls_ssl_certificate’
	Random Forest	11	‘qty_dot_url’, ‘qty_hyphen_url’, ‘qty_underline_url’, ‘qty_slash_url’, ‘qty_equal_url’, ‘length_url’, ‘qty_dot_domain’, ‘qty_vowels_domain’, ‘domain_length’, ‘qty_nameservers’, ‘qty_mx_servers’
	Gradient Boosting	17	‘qty_dot_url’, ‘qty_hyphen_url’, ‘qty_underline_url’, ‘qty_slash_url’, ‘qty_plus_url’, ‘qty_percent_url’, ‘qty_tld_url’, ‘length_url’, ‘qty_dot_domain’, ‘qty_hyphen_domain’, ‘qty_vowels_domain’, ‘domain_length’, ‘domain_in_ip’, ‘qty_nameservers’, ‘qty_mx_servers’, ‘tls_ssl_certificate’, ‘url_shortened’
GA	SVM	21	‘qty_dot_url’, ‘qty_hyphen_url’, ‘qty_underline_url’, ‘qty_slash_url’, ‘qty_equal_url’, qty_at_url’, ‘qty_and_url’, ‘qty_hashtag_url’, ‘qty_percent_url’, ‘length_url’, ‘qty_dot_domain’, ‘qty_hyphen_domain’, ‘qty_tilde_domain’, ‘qty_plus_domain’, ‘qty_vowels_domain’, ‘domain_length’, ‘domain_in_ip’, ‘server_client_domain’, ‘qty_nameservers’, ‘qty_mx_servers’, ‘tls_ssl_certificate’

- **Appendix 4: Models Evaluation without GA parametrization**

Classifier	Feature Selection Algorithm	Number of Features	Pre (%)	Recall (%)	F-Score (%)	Acc (%)
RF	None	45	91.5	91.7	91.59	94.18
	MI	30	91.12	90.4	90.75	93.63
	RFECV	13	91.45	91.64	91.54	94.15
	GA	21	91.59	91.89	91.74	94.27
XGBoost	None	45	88.88	89.68	89.28	92.55
	MI	30	88.73	89.59	89.15	92.46
	RFECV	13	89.09	89.59	89.34	92.61
	GA	21	88.66	89.54	89.1	92.43
NB	None	45	87.76	28.05	42.5	73.77
	MI	30	87.44	29.85	44.53	74.3
	RFECV	13	87.01	58.27	69.79	82.56
	GA	21	88.53	64.3	74.49	85.62
KNN	None	45	91.81	85.34	88.45	92.3
	MI	30	91.79	85.3	88.43	92.28
	RFECV	13	90.98	86.89	88.89	92.49
	GA	21	91.89	90.23	91.05	93.7
NN	None	45	85.2	90.7	87.9	91.66
	MI	30	86.21	88.51	87.4	91.32
	RFECV	13	85.94	88.20	87.02	91.04
	GA	21	85.83	91.2	88.43	91.73

- **Appendix 5: Models Evaluation with GA parametrization**

Classifier	Feature Selection Algorithm	Number of Features	Pre (%)	Recall (%)	F-Score (%)	Acc (%)
RF	None	45	91.54	92.07	91.8	94.4
	MI	30	91.46	91.88	91.66	94.22
	RFECV	13	91.92	91.62	91.77	94.30
	GA	21	92.56	92.78	92.67	95.34
XGBoosting	None	45	92.08	91.07	91.57	94.20
	MI	30	92.25	91.32	91.78	94.35
	RFECV	13	92.01	90.39	91.19	93.96
	GA	21	92.1	92.81	92.45	94.89
KNN	None	45	89.96	87.94	88.93	92.43
	MI	30	90.6	87.22	88.88	92.45
	RFECV	13	90.98	86.89	88.89	92.49
	GA	21	90.69	88.62	89.64	92.87
NN	None	45	86.71	91.54	89.05	92.54
	MI	30	86.95	92.32	89.55	92.61
	RFECV	13	87.41	89.9	88.64	92.87
	GA	21	88.76	92.31	90.5	93.5

- **Appendix 6:** Parameters settings of GASearchCV optimization for XGBoost with GA feature selection.

Parameter	Value
Estimator	GradientBoostingClassifier
Population size	30
Generations	20
Crossover probability	ExponentialAdapter(initial_value=0.05, end_value=0.8, adaptive_rate=0.1)
Mutation probability	ExponentialAdapter(initial_value=0.05, end_value=0.8, adaptive_rate=0.1)
Algorithm	eaMuCommaLambda
Evaluation	Cross validation with k=3

- **Appendix 7:** Parameters settings of XGBoost classifier with GA feature selection resultant from GASearchCV optimization.

Parameter	Value
Learning Rate	0.089
Maximum Depth	5
Estimators Number	901
Subsample	0.684

- **Appendix 8:** Parameters settings of GASearchCV optimization for KNN with GA feature selection.

Parameter	Value
Estimator	KNeighborsClassifier
Population size	30
Generations	20
Crossover probability	ExponentialAdapter(initial_value=0.05, end_value=0.8, adaptive_rate=0.1)
Mutation probability	ExponentialAdapter(initial_value=0.05, end_value=0.8, adaptive_rate=0.1)
Algorithm	eaCommaLambda
Evaluation	Cross validation with k=3

- **Appendix 9:** Parameters settings of KNN classifier (with GA feature selection) resultant from GASearchCV optimization.

Parameter	Value
Algorithm	brute
Leaf size	20
Neighbors number	3

- **Appendix 10:** Parameters settings of GASearchCV optimization for DNN with GA feature selection.

Parameter	Value
Estimator	Keras model
Population size	50
Generations	20
Crossover probability	ExponentialAdapter(initial_value=0.05, end_value=0.8, adaptive_rate=0.1)
Mutation probability	ExponentialAdapter(initial_value=0.05, end_value=0.8, adaptive_rate=0.1)
Algorithm	eaMuCommaLambda
Evaluation	Cross validation with k=3

- **Appendix 11:** Parameters settings of DNN classifier (with GA feature selection) resultant from GASearchCV optimization.

Parameter	Value
Batch Size	40
Epochs	10
Number of layers	4
Number of neurons	60
Dropout	0.1
Last Layer Activation	sigmoid
Optimizer	Adam
Losses	Binary crossentropy
Metrics	Accuracy
Activation function	Relu