A NEW MACHINE LEARNING CLASSIFICATION ALGORITHM FOR PHISHING URLS DETECTION

2023
Ph.D. THESIS
COMPUTER ENGINEERING

Abdalraouf Almahdi Mohammed ALARBI

Thesis Advisor
Assoc. Prof. Dr. Zafer ALBAYRAK

A NEW MACHINE LEARNING CLASSIFICATION ALGORITHM FOR PHISHING URLS DETECTION

Abdalraouf Almahdi Mohammed ALARBI

Thesis Advisor
Assoc. Prof. Dr. Zafer ALBAYRAK

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering

Prepared as
Ph.D. Thesis

KARABUK
June 2023

I certify that in my opinion the thesis submitted by Abdalraouf Almahdi Mohammed ALARBI titled "A NEW MACHINE LEARNING CLASSIFICATION ALGORITHM FOR PHISHING URLS DETECTION" is fully adequate in scope and quality as a thesis for the degree of Ph.D. of Science.

Assoc. Prof. Dr. Zafer ALBAYRAK .....
Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Doctoral of Science thesis. June 23,2023

Examining Committee Members (Institutions)	Signature
Chairman : Prof. Dr. Necmi Serkan TEZEL (KBU)	.....
Member : Assoc. Prof. Dr. Zafer ALBAYRAK (SUBU)	.....
Member : Assoc. Prof. Dr. Yüksel ÇELİK (KBU)	.....
Member : Assist. Prof. Dr. Muhammet ÇAKMAK (KBU)	.....
Member : Assist. Prof. Dr. Fatih VARÇIN (SU)	.....

The degree of Ph.D. Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Prof. Dr. Müslüm KUZU .....
Director of Graduate Education Institute

"I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well."

Abdalraouf Almahdi Mohammed ALARBI

ABSTRACT

Ph. D. Thesis

A NEW MACHINE LEARNING CLASSIFICATION ALGORITHM FOR PHISHING URLS DETECTION

Abdalraouf Almahdi Mohammed ALARBI

Karabuk University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assoc. Prof. Dr. Zafer Albayrak
June 2023, 96 pages

In today's era of ever-increasing online dangers, the identification of phishing URLs has become a critical task to ensure user safety and protect sensitive information. With the rise in sophisticated cyberattacks, hackers have become adept at creating deceptive websites that mimic legitimate ones, making it challenging for users to distinguish between genuine and fraudulent URLs. This has led to an urgent need for robust and advanced techniques to detect and mitigate the risks associated with phishing attacks. By employing advanced algorithms and machine learning models, cybersecurity experts are continuously working towards enhancing the accuracy and efficiency of phishing URL detection systems, empowering users to make informed decisions while navigating the vast digital landscape.

The study in this thesis consists of three stages. In the first stage, we propose a new classification algorithm called the Core Classification Algorithm (CCA), which is

derived from the K-nearest neighbor algorithm (KNN) and hybridized with the unsupervised algorithm K-means. The primary objective is to find similarities while overcoming the challenge of excluding non-representative cores from the clusters. The hybridization process aims to leverage the synergies created by combining two different algorithms, iteratively modifying outcomes to achieve optimal solutions. This strategy improves the efficiency and accuracy of classifying data into two or more clusters based on their labels.

In the second stage, we introduce the Improved Core Classification Algorithm (ICCA), an adaptation of the algorithm used in the previous section. Instead of relying on a single core point, we employ active sets. Compared to the utilization of various other available algorithms, this approach yields more accurate results.

Finally, we analyzed phishing URLs using a comprehensive dataset consisting of 549,346 entries. Among these entries, 392,897 URLs were identified as phishing attempts, while 114,299 URLs were classified as legitimate. We conducted several preprocessing steps, including data cleaning, feature engineering, and feature selection, to enhance the overall quality of our analysis. These processes provided us with in-depth insights into the data and allowed us to extract critical features. Subsequently, we evaluated our algorithms, and the findings demonstrated encouraging prediction accuracy.

Keywords : Classification; Phishing attacks; K-means; Hybridization; Core point; Active set; Clustering.

Science Code : 92403

ÖZET

Doktora Tezi

KİMLİK AVI URL TESPİT İÇİN YENİ BİR MAKİNE ÖĞRENİMİ SINIFLANDIRMA ALGORİTMASI TASARIMI

Abdalraouf Almahdi Mohammed ALARBI

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Doç. Dr. Zafer ALBAYRAK
Haziran 2023, 96 sayfa

Günümüzde çevrimiçi tehlikelerin sürekli artmasıyla, kimlik avı URL'lerini belirlemek, kullanıcıların güvenliğini sağlamak ve hassas bilgileri korumak gittikçe daha önemli bir görev haline gelmektedir. Bu tezde bu problemlere çözüm olarak, Çekirdek Sınıflandırma Algoritması (CCA) adını verdiğimiz yeni bir sınıflandırma algoritması önerilmiştir. Bu algoritma, K-means algoritması ile hibritlenerek türetilmiştir. Hibritleştirme sürecinin amacı, mümkün olan en iyi çözümlere ulaşmak için sonuçları yinelemeli olarak değiştirerek iki farklı algoritma birleştirilmiştir. Bu strateji, verilerin bu kümelere sınıflandırılma doğruluğunu artırmanın yanı sıra, etiketleri temel alarak iki veya daha fazla kümeye ayırarak sınıflandırma verimliliğinin arttırılması sağlanmıştır.

Tezin sonraki bölümünde, bir önceki bölümde kullanılan algoritmanın bir uyarlaması olan Enhanced Core Classification Algorithm (ICCA) sunulmuştur. Bu yinelemede tek

bir çekirdek noktaya güvenmek yerine, bunun yerine Aktif kümeler kullanılmıştır. Literatürdeki diğer çeşitli algoritmalar ile karşılaştırıldığında, bu yöntemin sonuçlarının literatürdeki diğer algoritmalardan daha iyi sonuçlar verdiği görülmüştür.

Tezin son bölümünde, içinde 549.346 giriş bulunan kapsamlı bir veri kümesini kullanarak kimlik avı URL'leri hakkında bir analiz yapmıştık. Bu girişler arasında phishing girişimi olduğu tespit edilen 392.897 URL ve yasal kabul edilen 114.299 URL vardı. Analizimizin genel kalitesini iyileştirebilmek için veri temizleme, özellik mühendisliği ve keşif veri analizi (EDA olarak da bilinir) gibi bir dizi ön işleme adımı gerçekleştirdik. Bu süreçler sayesinde, verilere ilişkin daha derinlemesine içgörüler elde edebildik ve kritik öneme sahip özellikleri ayıklayabildik. Ardından algoritmalarımızın analizini yaptık ve elde ettiğimiz bulgular tahminlerinin doğruluğu açısından cesaret vericiydi.

Anahtar Sözcükler : Sınıflandırma; Kimlik avı saldırıları; K-anlamı; Hibridizasyon; Çekirdek nokta; Aktif küme; Kümeleme.

Bilim Kodu : 92403

ACKNOWLEDGMENT

There are no words to express my heartfelt thanks and admiration for my thesis and research adviser, Assoc. Dr. Zafer ALBAYRAK. He has motivated me to become an independent researcher and has shown me the value of critical thinking. He also showed what a clever and hardworking scientist can do.

My heartfelt gratitude also goes to the members of my thesis advising and examination committee: Prof. Dr. Necmi Serkan TEZEL and Assoc. Prof. Dr. Yüksel ÇELİK. They gladly provided their time to provide me with constructive feedback on my work. Their great knowledge and expertise have encouraged me throughout my studies.

I am grateful to my parents for their unwavering trust, timely encouragement, and unending patience. When I was tired, it was their affection that helped me get back up. Finally, I express my heartfelt gratitude to my wife and sons. My wife, who has been my closest friend and terrific companion, loving, supporting, encouraging and helping me get through this difficult moment in the most positive manner.

CONTENTS

Page

Page

LIST OF FIGURES

Page

Page

LIST OF TABLES

Page

SYMBOLS AND ABBREVIATIONS INDES

ABBREVITIONS

D_M	: Distance matrix.
C	: Class.
Dsc	: The number of points in each class
(Dm)c1	: Distance matrix for each class
(cor v)c	: Core vector
Core_v)c	: The core point which has highest connectivity in the class
A_S	: Active set
B	: The number of objects in the active set
url_length	: Length of URL
qty_dot_url	: Quantity of (.) in URL
qty_hyphen_url	: Quantity of (-) in URL
qty_slash_url	: Quantity of (/) in URL
qty_questionmark_url	: Quantity of (?) in URL
qty_equal_url	: Quantity of (=) in URL
qty_at_url	: Quantity of (@) in URL
qty_and_url	: Quantity of (&) in URL
qty_exclamation_url	: Quantity of (!) in URL
qty_space_url	: Quantity of ( ) in URL
qty_tilde_url	: Quantity of (~) in URL
qty_comma_url	: Quantity of (,) in URL
qty_plus_url	: Quantity of (+) in URL
qty_asterisk_url	: Quantity of (*) in URL
qty_hashtag_url	: Quantity of (#) in URL

qty_dollar_url : Quantity of (\$) in URL

qty_percent_url : Quantity of (%) in URL

domain_length : Length of domain

qty_dot_domain : Quantity of (.) in domain

qty_hyphen_domain : Quantity of (-) in domain

path_length : Length of path

qty_dot_path : Quantity of (.) in path

qty_hyphen_path : Quantity of (-) in path

qty_slash_path : Quantity of (/) in path

qty_equal_path : Quantity of (=) in path

qty_at_path : Quantity of (@) in path

qty_and_path : Quantity of (&) in path

qty_exclamation_path : Quantity of (!) in path

qty_space_path : Quantity of ( ) in path

qty_tilde_path : Quantity of (~) in path

qty_comma_path : Quantity of (,) in path

qty_plus_path : Quantity of (+) in path

qty_asterisk_path : Quantity of (*) in path

qty_dollar_path : Quantity of (\$) in path

qty_percent_path : Quantity of (%) in path

query_length : Length of query

qty_dot_query : Quantity of (.) in query

qty_hyphen_query : Quantity of (-) in query

qty_slash_query : Quantity of (/) in query

qty_questionmark_query : Quantity of (?) in query

qty_equal_query : Quantity of (=) in query

qty_at_query : Quantity of (@) in query

qty_and_query : Quantity of (&) in query

qty_exclamation_query : Quantity of (!) in query

qty_space_query : Quantity of ( ) in query

qty_tilde_query : Quantity of (~) in query

qty_comma_query : Quantity of (,) in query

qty_plus_query : Quantity of (+) in query

qty_asterisk_query : Quantity of (*) in query

qty_dollar_query : Quantity of (\$) in query

qty_percent_query : Quantity of (%) in query

fragment_length : Length of fragment

qty_dot_fragment : Quantity of (.) in fragment

qty_hyphen_fragment : Quantity of (-) in fragment

qty_slash_fragment : Quantity of (/) in fragment

qty_questionmark_fragment : Quantity of (?) in fragment

qty_equal_fragment : Quantity of (=) in fragment

qty_and_fragment : Quantity of (&) in fragment

qty_exclamation_fragment : Quantity of (!) in fragment

qty_space_fragment : Quantity of ( ) in fragment

qty_comma_fragment : Quantity of (,) in fragment

qty_asterisk_fragment : Quantity of (*) in fragment

qty_hashtag_fragment : Quantity of (#) in fragment

qty_dollar_fragment : Quantity of (\$) in fragment

qty_percent_fragment : Quantity of (%) in fragment

DEA : Data Envelopment Analusis

PART 1

INTRODUCTION

During the era of increased Internet usage and a rise in Phishing URLs worldwide, researchers have long been employing machine learning (ML) techniques to protect individuals. Numerous studies have shown promising results, although attackers have become more sophisticated in detecting and rectifying errors. The emergence of the Internet as a primary medium for business and personal communication has led to the emergence of crucial research areas concerning online credibility and illicit activities [1].

Classification is the process of dividing a dataset based on its labels. All classification methods follow a two-step approach: firstly, a model is trained to categorize the dataset into two or more groups; secondly, the model, usually represented by a mathematical formula, is evaluated on an unseen dataset to determine its performance, which determines its acceptance or rejection. Support Vector Machines (SVM), Naive Bayes (NB), Decision Trees (DTs), and various other algorithms are widely used for classification [2].

Machine learning (ML) and classification find applications in various sectors such as manufacturing, retail, healthcare, and life sciences, among others. Machine learning plays a pivotal role in bolstering cybersecurity by offering advanced and dynamic defense mechanisms against evolving threats. By analyzing vast amounts of data, machine learning algorithms can detect patterns, anomalies, and potential vulnerabilities that may go unnoticed by traditional security systems. Through continuous learning and adaptation, these algorithms improve the accuracy and efficiency of intrusion detection, malware identification, and user behavior analysis. Furthermore, machine learning empowers cybersecurity experts to automate processes, optimize resource allocation, and respond swiftly to potential breaches,

mitigating the risks and potential damage associated with cyberattacks. Ultimately, integrating machine learning into cybersecurity enables organizations to fortify their defenses, enhance proactive threat detection, and safeguard critical digital assets in an increasingly complex and interconnected digital landscape [2, 3]. Phishing attacks pose a significant threat to individuals and businesses alike, making it crucial for programmers to focus on detecting and preventing these attacks. The aim is to minimize opportunities for hackers to steal sensitive information such as personal and bank account passwords, telecommunications records, and business data [4]. Phishing victims are deceived into providing private information by being directed to websites that closely resemble the ones they typically use. The prevalence of phishing scams is rapidly increasing on an international scale [5]. One major issue is the proliferation of social engineering techniques that mimic URLs and websites to extract various types of user data, including personal information, bank account details, and passwords. The development of effective methods to detect phishing URLs is a critical aspect of addressing this problem, especially considering the potential risks to enterprises and their sensitive data [6]. This discussion would also cover different approaches to classification algorithms in machine learning (ML), emphasizing their applications in medical research, predictions, and healthcare data processing. Classification algorithms have found extensive use in healthcare, phishing attack detection, business, finance, and other domains. Accurate predictions and unambiguous categorization are crucial in ML, and data scientists employ various algorithms and models to extract patterns that generate actionable insights [7]. The process of studying data aids in training the classifier to better understand the dataset. Classification proves most beneficial when predicting certain attributes based on training, such as determining a person's gender or identifying high blood sugar levels. However, in fields where erroneous forecasts are heavily discouraged, the topic of predictions can be sensitive. Several classification algorithms, including Support Vector Machines (SVM), Naive Bayes, Decision Trees, and Neural Networks, have shown superior performance in diagnosing diseases, leveraging data mining and ML techniques to handle large volumes of data from diverse sources. Evaluating the accuracy of these algorithms against each other is essential [8]. Each algorithm has its own mechanism or strategy for constructing suitable models, ranging from probabilistic approaches to neural networks and linear equations for weight updates. K-nearest neighbors (KNN) relies

on the nearest neighbor strategy, while SVM employs linear algebra. Through extensive research on various classification algorithms documented in published literature, it was found that artificial neural network (ANN) algorithms generally apply to all data domains, including audio, images, video, text, and standard datasets. Despite this, the availability of numerous algorithms remains advantageous as each algorithm possesses unique characteristics to tackle different dataset challenges. Some algorithms excel in certain cases while being deficient in others, and vice versa.

However, due to several reasons, primarily the extensive nature of medical processes, it will take a considerable amount of time before AI replaces human professionals. While the potential of ML offers the means to automate certain aspects of therapy, there are significant challenges impeding its rapid adoption in healthcare. This stands as one of the key obstacles in the field. Deep learning, also known as neural network models with multiple layers of features or variables, can provide accurate predictions for complex datasets such as X-rays, cloud architecture, graphs, and images, which may contain numerous hidden features requiring analysis. Since the 1970s, when MYCIN was developed to treat blood-borne and bacterial infections, ML has primarily focused on diagnosing and treating various disorders. However, these systems were unable to integrate with clinicians' workflows or medical record systems, and they could not replace human diagnosticians [9].

This thesis introduces a novel classification algorithm called the Core Classifier Algorithm (CCA), which is based on cores representing the distinctive attributes and traits of each category. These cores are employed to classify new data points based on their resemblance to the cores. Each class is characterized by its unique features. The CCA incorporates the K-means clustering method to emulate the learning mechanism of neural networks and mitigate the negative effects of anomalies in data distribution, such as outliers and overlaps. While the results of the K-means algorithm are not deterministic and depend on specific implementations, the main motivation for its use in the CCA is to generate centroids and improve model accuracy. The CCA methodology involves multiple iterations of K-values to effectively capture diverse distributions, converging towards the most optimal representation and achieving high levels of accuracy, even when the data exhibits significant variability. This is

analogous to neural networks, which undergo multiple iterations of weight adjustment and error rate computation for each feature until they reach optimal outcomes. The use of multiple iterations of K-values in the CCA enhances the representation of diverse distributions.

In the second stage of this study, the ICCA algorithm is presented as a derivative of the CCA algorithm. The ICCA algorithm enhances accuracy through the utilization of Active Set techniques, which play a crucial role in its effectiveness. Experimental results demonstrate the algorithm's efficacy across various domains, placing it on par with other established algorithms.

Finally, in this thesis, we apply preprocessing techniques to extract features from URLs and employ feature engineering and selection to train our model and detect phishing URLs using our algorithms. We compare the performance of our algorithm with other well-known algorithms such as SVM, Decision Trees, and Random Forest by measuring results using the confusion matrix. Our algorithm demonstrates high accuracy and good performance in this evaluation.

The thesis is organized as follows: Part 2, Literature review of machine learning, phishing attacks, and hybrid algorithms. Part 3, Theoretical background of machine learning, discussion of some algorithms used in the thesis, and an overview of the Phishing URLs domain. Part 4, Methodology employed in the thesis. Part 5, Presentation and discussion of the results. Part 6, Conclusion.

PART 2

LITERATURE REVIEW

2.1. MACHINE LEARNING AND PHISHING ATTACKS

In this study, the authors focused on a content-based three-stage series attack as a mechanism for phishing attacks. The model incorporated three variables: URLs, web traffic, and web content, aiming to identify factors contributing to phishing attack success or failure. To implement the proposed phishing attack method, a dataset of recent phishing attacks was compiled. Real phishing cases demonstrated higher accuracy in detecting both zero-day phishing attacks and common phishing attempts. The accuracy of phishing detection was assessed using three classifiers: Neural Network (NN), Support Vector Machine (SVM), and Random Forest (RF). The NN classifier achieved 95.18% accuracy, SVM achieved 85.45% accuracy, and RF achieved 78.89% accuracy. These findings highlight the effectiveness of employing machine learning in identifying phishing attack [10].

The authors aimed to enhance phishing detection accuracy by examining the utilization of email body language in their literature review. They found that email body text contains concealed information, justifying their endeavor. Accordingly, they propose a novel classifier leveraging natural language processing (NLP), deep learning techniques, and a Graph Convolutional Network (GCN) to identify phishing emails. This classifier analyzes the email content, utilizing NLP and deep learning algorithms to identify phishing characteristics. Its performance is evaluated using accuracy, precision, and recall metrics, and compared to state-of-the-art models. The proposed classifier demonstrates excellent performance when applied to a well-balanced and labeled dataset [11].

Given the increasing prevalence of cybercrime victimization, there is an urgent need for an intelligent defense mechanism to protect users. The inadequate adoption of security technologies is identified as the primary factor driving this surge. Deep learning has emerged as a significant advancement, surpassing traditional signature-based and classical machine learning approaches, due to its exceptional performance and comprehensive problem-solving capabilities. The rapid progress in deep learning techniques has facilitated this advancement. In this paper, authors introduce the LSTM, CNN, and LSTM-CNN algorithms as effective approaches for distinguishing and categorizing website URLs as genuine or phishing. The evaluation of this proposed solution demonstrates highly favorable outcomes in identifying phishing websites. However, these recommended deep learning algorithms exhibited considerable variability in performance when applied to the same dataset [12].

The objective of this study is to propose a framework utilizing the stacking model for the detection of phishing websites [13]. Phishing is a scam where criminals steal user credentials to make money. Cybercrime impacts e-commerce, internet business, banking, and digital marketing. Phishers use spam emails and fake websites that seem real. Targeted websites steal consumers' personal data. Information gain, gain ratio, Relief-F, and recursive feature elimination (RFE) are used to evaluate phishing datasets. Two qualities are created from the strongest and weakest. RF, NN, bagging, support vector machine, Naive Bayes, and k-nearest neighbor are used for principal component analysis on the chosen and remaining features. Next, two stacking models, Stacking1 (RF NN Bagging) and Stacking2 (KNN RF Bagging), combine the best classifiers to improve the proposed features and all classifiers. RFE successfully removes the dataset's least significant features. Stacking1 (RF NN Bagging) detects and classifies phishing websites better than other classifiers.

Website vulnerabilities to malicious attacks are examined in this article. Machine learning improves predictions. Phishing assaults and botnets have increased in recent years. The authors threats exploit deceptive URLs to fool visitors. Decision tree and logistic regression methods are used to handle real-time difficulties and predict end user concerns. The information comprises 420,000 legitimate and affected websites.

Testing datasets assess prediction time and accuracy. Logistic regression improves efficiency and accuracy [14].

This study introduces hybrid deep learning models designed to detect phishing uniform resource locators (URLs). These models leverage long short-term memory and deep neural network methods. The evaluation of these models is conducted using datasets specific to phishing. The proposed hybrid deep learning models incorporate character embedding and natural language processing (NLP) features. By incorporating these features, the models are able to effectively utilize both the deep connections between characters and the high-level connections based on NLP. The experimental results demonstrate that the suggested models outperform other existing phishing detection models in terms of accuracy [15].

2.2. HYPER ALGORITHMS

Numerous articles have been published describing one of the two primary approaches to solving this problem. When training data are available, supervised methodologies utilize machine learning algorithms. When linguistic resources are available, an unsupervised method based on a semantic orientation is utilized. Few studies, however, integrate the two approaches. The authors of this paper propose using meta-classifiers that combine supervised and unsupervised learning to construct a polarity classification system. Researchers have utilized a Spanish corpus of film evaluations alongside its parallel corpus in English. Initially, two distinct models are generated using these two corpora and machine learning algorithms. By integrating SentiWordNet into the English corpus, a new unsupervised model is generated. The three systems are combined using a meta-classifier that permits the application of multiple combination algorithms, such as the voting system or layering. When authors work with parallel corpora, the results obtained are superior to those obtained using the systems individually, indicating that this approach may be a viable strategy for polarity classification [16].

Semi-automatic and automatic MS plaque identification, segmentation, and classification technologies have increased in recent years. This research presents an

automatic mixed method using a typical unsupervised machine learning algorithm and a deep-learning attention-gate 3D U-net network. The deeplearning network is trained to segment infratentorial and juxtacortical plaques in clinical MRIs, which the standard technique struggles with. It was trained and validated using a multi-center multi-scanner dataset of 159 cases with T1 weighted (T1w) and FLAIR images and hand MS plaque delineations segmented and validated by a panel of raters. Lesion-wise Dice score measured detection. Combining the two pipelines' output segmentations requires a simple label fusion. This integrated strategy detects infratentorial and juxtacortical lesions 14% and 31% better than the unsupervised machine learning pipeline utilized as a performance assessment baseline [17].

This study presents a brief comparison of the proposed model with commonly used machine learning models including AdaBoost, XGBoost, Random Forest, Gaussian Naive Bayes, and LGB. The purpose of this comparison is to illustrate the strengths and weaknesses of the suggested model. In the context of network intrusion traffic detection, the experimental results demonstrate that the accuracy level of their developed model is approximately 11% higher than that of previous models [18].

The proposed method combines diverse agents to improve the accuracy of predictions. In particular, supervised learning, which offers a direct The authors propose applying unsupervised exploratory methods to the data set to obtain a better understanding of the data's quality. This enhances the selection and categorization of data for creating training sets prior to machine learning application. Researchers demonstrate this using a genome-wide small interfering RNA screen with a high content. They conduct an unsupervised exploratory data analysis to facilitate the identification of four robust phenotypes, which they then use as a training set to construct a high-quality random forest machine learning model capable of differentiating four phenotypes with a 91.1% accuracy and a kappa of 0.85. In comparison to the use of unsupervised methods alone, their approach improved their ability to extract new information from the display [19].

This research presents a novel hybrid strategy that uses Bayesian optimization (BO) and a modified GA-PARSIMONY algorithm to generate parsimonious models. This method reduces computational complexity, which limits GA-PARSIMONY. Bayesian

optimization, often known as Bayes' theorem, is used to find good model parameters to overcome this restriction. After that, a limited iteration of the Genetic Algorithm-PARSIMONY produces correct parsimony models. For accuracy, the approach uses feature reduction, data transformation, and parsimonious model selection. The hybrid technique is tested on 10 UCI datasets using extreme gradient boosting machines (XGBoost). The hybrid method yields models equivalent to GA-PARSIMONY. The hybrid technique decreases processing time for eight of the 10 datasets, demonstrating its efficiency. This study introduces a hybrid technique for creating parsimonious models and shows the possibilities of merging Bayesian optimization with GA-PARSIMONY. The study solves computational complexity and creates accurate models in less time by integrating these two techniques. Experiments on varied datasets show the hybrid technique's efficacy and potential. This method can improve model derivation in many disciplines. This hybrid strategy may be used to more model optimization and complexity reduction problems in future study [20].

The proposed approach brings together different kinds of agents in order to improve the accuracy of predictions. Specifically, supervised learning, which provides a direct mapping between the data domain and the solution domain while simultaneously introducing bias to generalize the mapping, is combined with unsupervised learning, which does not depend on similar generalization bias or training data but also does not provide a direct mapping between the data and solution domains. This results in a more accurate mapping between the data domain and the solution domain than would be possible using supervised learning alone. The combination is achieved by the utilization of the joint probability density function (PDF) of the supervised classification. This function is put to use in order to direct the identification of clusters that have been demarcated by unsupervised learning. This multi-agent strategy can limit the amount of bias that is introduced during training, and it also offers a foundation for the generation of a probability distribution for each sample rather than a discrete classification. In turn, the distribution can be utilized to more properly describe the continuous character of well log signals, which reflects continuity in lithological regimes. This, in turn, allows for greater precision [21].

The authors of this work explore the creation of a hybrid algorithm that incorporates two supervised algorithms, Naive Bayes and C4.5 as it shows in the Figure 2.1, to enhance the training process of network intrusion detection models, specifically focusing on SDN. By integrating the label field of each data sample during learning and training, the algorithm achieves improved training results with enhanced performance. Furthermore, the hybrid algorithm efficiently reduces the computational burden by consolidating the calculation of gain values into a single process, thereby minimizing unnecessary time expenditure. This reduction in time is particularly significant since the calculation of gain values necessitates referencing the entire training dataset. The findings from experimental evaluations underscore the practical advantages of the proposed algorithm, as it not only reduces the required training time but also enhances the overall performance of intrusion detection, surpassing other existing hybrid algorithms [22].

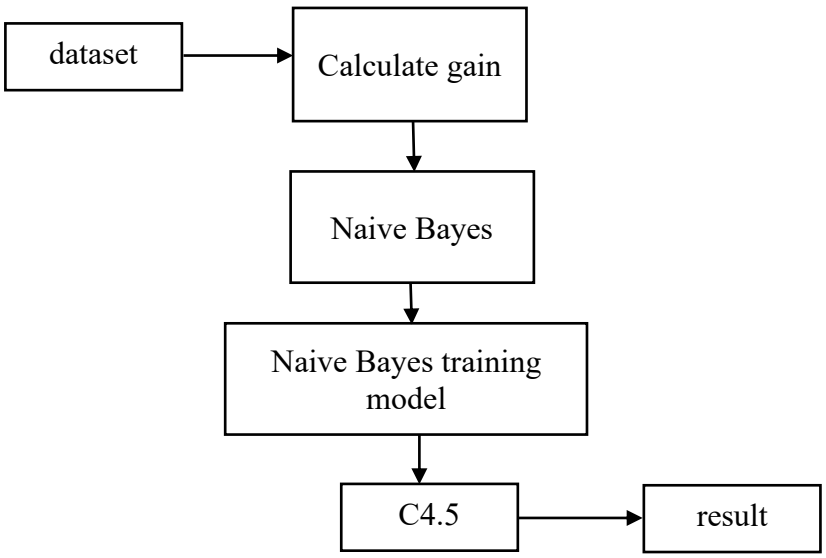

Figure 2.1.The architecture of our proposed hybrid algorithm [22].

Further research can expand on this work by exploring the optimization of the hybrid algorithm's parameters and evaluating its applicability to different types of intrusion detection datasets. Additionally, investigations into the interpretability and robustness of the hybrid model could provide valuable insights into its practical implementation. Ultimately, this novel hybrid machine learning approach holds promise for enhancing

the effectiveness and efficiency of network intrusion detection systems, contributing to the advancement of cybersecurity [22].

In any distributed system, one of the most challenging tasks is to ensure network protection by identifying various attack scenarios. Intrusion detection systems have gained popularity for examining and identifying network attacks to enhance the security of data transmission. This paper focuses on developing a hybrid anomaly-based intrusion detection model that combines two machine learning algorithms, each compensating for the limitations of the other, resulting in strong performance and a high detection rate. Specifically, the paper explores how each algorithm addresses the weaknesses of the other. The Random Forest algorithm is employed for feature selection, while the Classification and Regression Trees algorithm is utilized for classification. Both algorithms are utilized in this study [23].

Moreover, the methodology employed in this study has the potential to be generalized beyond OSA, as it can be adapted to identify high-risk patient groups in other complex and diverse disorders. By leveraging similar multimetric approaches, healthcare professionals can enhance their understanding of various diseases and tailor interventions to specific patient populations. Future research directions could involve validating the multimetric phenotyping framework using larger and more diverse datasets to establish its robustness and generalizability. Additionally, exploring the application of the framework in clinical settings and assessing its impact on patient outcomes would provide further evidence of its utility and effectiveness. Nevertheless, the integration of supervised and unsupervised machine learning techniques in the multimetric phenotyping framework offers a comprehensive approach to classify OSA patients. This approach reduces subjectivity, improves reliability, and reveals novel subgroups with distinct risks of developing associated disorders. The potential application of the framework extends beyond OSA, making it a valuable tool for identifying high-risk patient groups in various complex diseases [24].

The authors present a paradigm for semi-supervised learning that integrates clustering and classification into a single concept. Clustering analysis is a potent knowledge-discovery method, and it has the potential to uncover the underlying data space

structure from unlabeled data as shown in Figure 2.2. This is what motivates the researchers to do the study. To assist in the development of a more accurate classifier, our system incorporates semi-supervised clustering into the self-training classification process. Clustering is done with a semi-supervised fuzzy c-means technique, while classification is done with support vector machines. Both of these algorithms are employed, respectively. The benefits of the suggested framework have been demonstrated through experiments conducted on both simulated and actual datasets [25].

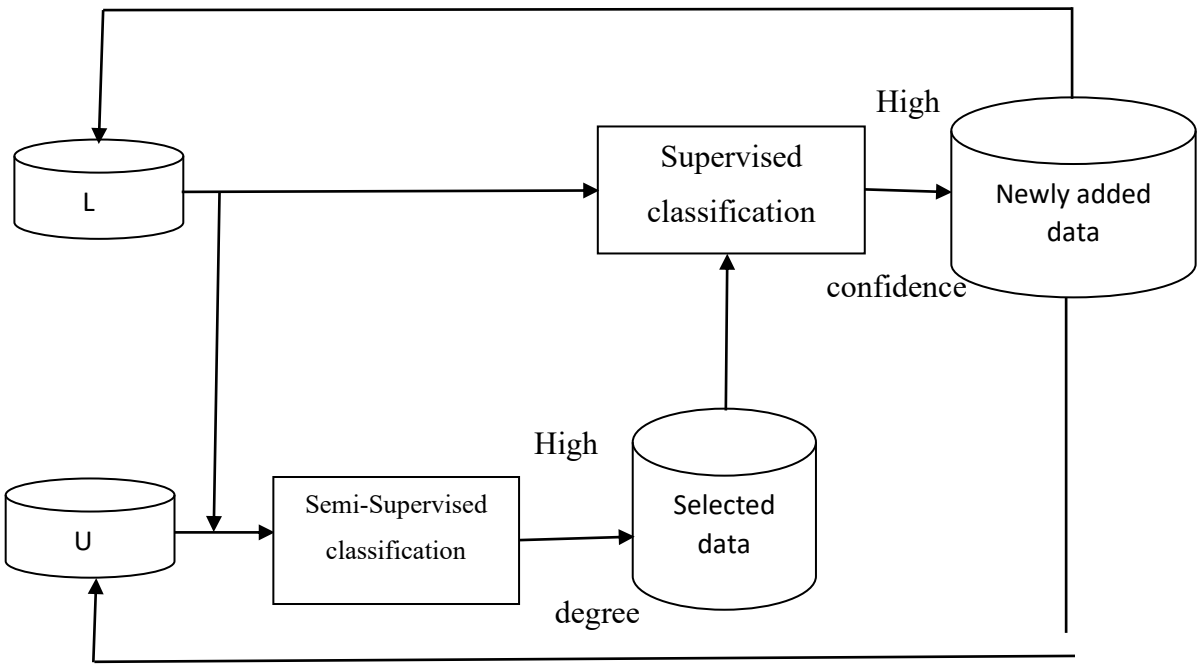

Figure 2.2. Flow chart of the framework (L and U represent labeled and unlabeled data, respectively) [25].

The primary aim of this study is to examine the existing data mining methods utilized for clustering and explore novel approaches to enhance clustering accuracy. The specific objective of this research is to develop an advanced clustering algorithm that builds upon an existing method. This study presents a distinctive approach that combines spectral clustering, k-means, and NFPH. The conventional initialization technique for cluster centroids in traditional k-means algorithms is substituted with the proposed system, as depicted in. This method targets the initial centroid to overcome k-means algorithm restrictions. The most relevant centroid for the situation is chosen rather than randomly. The suggested approach is trained utilizing publicly available

medical test datasets for study. WEKA, an open-source data mining software, evaluates it. The method is evaluated on ten University of California, Irvine datasets. Clustering error decreased by 2% and processing time increased from 4 to 5 seconds. The new k-means initialization strategy caused this processing delay. The system also reduces spectral clustering error. This technology improves accuracy but takes 4 seconds to process as it can be seen in the Figure 2.3 [26].

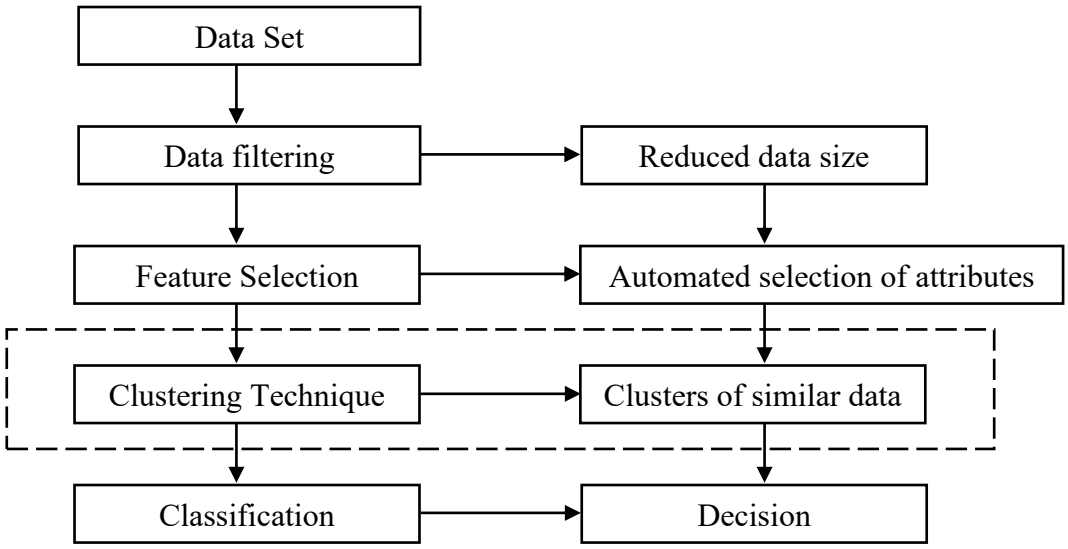

Figure 2.3. stat of art work flow [26].

In this study, the authors introduce a novel classification framework that incorporates a distinct ensemble classification step after the ensemble clustering stage. The objective of this framework is to specifically identify patients who have not been clustered, as depicted in Figure 2.4 [27].

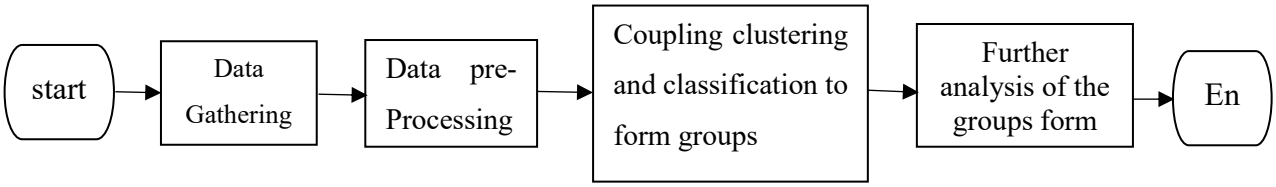

Figure 2.4. Abstract flowchart of the steps in the proposed pipeline [27].

Hence, a systematic procedure is developed that establishes a connection between ensemble clustering and ensemble classification, aiming to identify core groups, analyze data distribution within those groups, and enhance the final classification outcomes by addressing unclustered data. Subsequently, the proposed pipeline is

applied to a newly acquired real-world breast cancer dataset, followed by an assessment of its robustness and stability using standard datasets. The results demonstrate that adopting the described structure enables the generation of more accurate categorizations. Additionally, the findings are validated through the application of statistical tests, visualization techniques, evaluations of cluster quality, and insights from clinical experts [27].

The article proposes a clustering technique that combines PAM (Partitioning Around Medoids) and FCM (Fuzzy C-Means) to accurately classify faulty data. The approach first employs PAM to establish cluster prototypes, reducing the initial randomness of FCM. Subsequently, FCM is used to obtain the final clustering results. These measures are expected to enhance the algorithm's accuracy and require fewer iterations. To test and validate the effectiveness and efficiency of the proposed method, experiments are conducted using datasets related to faults in electrical equipment. The study's findings demonstrate that the combination of approaches employed in this work outperforms traditional methods of data analysis, such as the hierarchical clustering algorithm, in terms of both accuracy and computational efficiency [28].

This research study proposes an accurate and practical method for identifying printed ancient books. To reduce the error rate, a combination of sub-word clustering and an LSTM (Long Short-Term Memory) neural network is utilized as a character recognizer. Since limited information is available about the various font faces, researchers manually annotate certain sections of the books. The methodology involves clustering each sub-word in the book, followed by training an LSTM neural network using the manually labeled cluster centers. Finally, the clustering and classification results are combined to enhance the recognition rate [29].

In this study, the authors classified EEG signals using a classification model based on MLPNN (Multilayer Perceptron Neural Network). Using DWT (Discrete Wavelet Transform), the EEG signals were broken down into subbands. Instead of relying on fundamental statistical measures across the wavelet coefficients, the authors clustered the wavelet coefficients within each sub-band using the K-means algorithm. This method allowed for a more efficient analysis of the data. As depicted in Figure 2.5, the

probability distributions derived from the distribution of wavelet coefficients to the clusters were then used as inputs to the MLPNN model. Classification of data was accomplished with the MLPNN model. Five separate experiments were conducted to evaluate the performance of the proposed model in classifying distinct segments. These investigations included healthy and epileptic seizure-free segments, epileptic seizure segments, healthy segments, and epileptic seizure-free and epileptic seizure segments. The results demonstrated that the proposed model was effective in classifying the various tasks accurately. Therefore, the study's authors believe that the proposed model has the potential to be used as a diagnostic decision support mechanism in the management of epilepsy patients. [30].

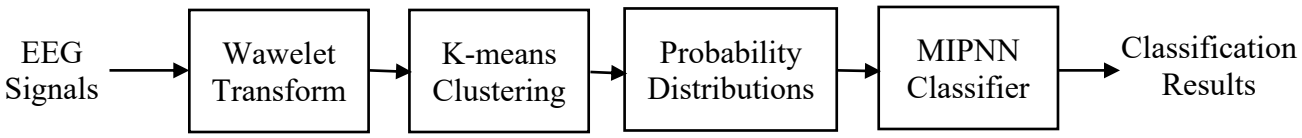

Figure 2.5. Schematic illustration of the proposed method [30].

The authors of this paper showcased the effectiveness of empirical risk minimization (ERM) as a method for selecting the next instance to label. However, ERM requires significant computational time. In the case of graphical data, researchers can employ graph topological analysis to swiftly identify instances that are likely to be suitable for labeling, enabling faster progression through the data. In this study, a novel approach is presented for identifying the best adjacent instance to a label, utilizing a metric based on clustering coefficients. Experimental results conducted on a dataset comprising 20 newsgroups and three binary classification tasks demonstrate that the utilization of clustering coefficient technique achieves comparable performance to ERM while significantly reducing computational time [31].

This work introduces a novel technique called EC3, which integrates clustering and classification for binary and multi-class classification tasks. EC3 utilizes an optimization function and combines multiple classification and clustering algorithms in a systematic manner. The authors theoretically establish the convexity and optimality of the problem, and solve it using the block coordinate descent technique. Furthermore, a variant of EC3 called iEC3 is proposed to handle imbalanced datasets.

Extensive experimental analysis is conducted, comparing EC3 and iEC3 with 14 baseline methods, including standalone classifiers, homogeneous ensemble classifiers, and heterogeneous ensemble classifiers that combine classification and clustering. The evaluation is performed on 13 standard benchmark datasets. The results demonstrate that both EC3 and iEC3 outperform the alternative baselines across all datasets, achieving at least a 10% higher AUC. Additionally, the suggested approaches exhibit faster execution compared to the best heterogeneous baseline method (1.21 times faster), increased robustness to noise and class imbalance, and improved accuracy compared to the best baseline method [32].

The authors have proposed a method for automatically identifying the learning style based on the extant behaviors of learners and using web usage mining techniques and machine learning algorithms. Utilizing web utilization mining techniques, the log file extracted from the E-learning environment was preprocessed and the sequences of the learners were captured. Based on the Felder and Silverman learning style model, the captured sequences of learners were input into the K-modes clustering algorithm to classify them into 16 learning style combinations. The naive Bayes classifier was then used to predict a student's learning approach in real time. The authors of the study used an actual dataset extracted from the log file of an e-learning system and the confusion matrix method to evaluate the performance of the employed classifier. The obtained results demonstrate that our strategy produces outstanding outcomes [33].

This paper proposes an unsupervised supervised machine learning approach, hierarchical clustering, and artificial neural network (ANN) by adopting a combined unsupervised-supervised method, unsupervised cluster analysis, and various supervised machine learning algorithms, such as Boostings, Support Vector Machine (SVM), and RReliefF. Researchers provide evidence that each cluster has its own foundation variables to predict, and Boosting and ANN estimation provide a more efficient framework for reducing the reserve error of insurers. Also, the different value and order of RReliefF between Boosting and OLS indicate an under- or over-estimated predictor, and the consistency of each year's influential variables over time indicates that the firm's loss reserve model from the previous year can predict the future loss reserve error. This article contributes to the existing literature by proposing a more

robust, consistent, and efficient prediction method (i.e., the unsupervised-supervised combination method) to enhance the loss reserve error prediction of insurers [34].

As indicated in the literature study, various algorithms have been employed for classification, yielding different results. However, many studies have utilized a combination of supervised and unsupervised algorithms, which has proven to be effective in improving classification outcomes. The use of machine learning in detecting phishing URLs has been extensively explored, but the results have shown significant variability. Consequently, enhancing existing algorithms to achieve better results has become a major focus for researchers, given the increasing sophistication of frauds.

The contributions of this thesis are as follows:

- Simulating the K-nearest neighbors (KNN) algorithm and identifying one core for each class, bearing its unique characteristics, instead of altering the classification outcome based on the K-value in KNN.
- Utilizing a clustering algorithm to address dataset distribution issues such as nonlinear classification, overlapping, or noise. These problems can be simulated through hidden layers in neural networks (NNs).
- Considering the instability of results in K-means, multiple iterations are required to construct different numbers of clusters, resulting in newer cores. This approach aligns with the methodology employed in NNs.
- Incorporating an updated version of the CCA algorithm called ICCA, which relies on an active set (A_S) to provide a more accurate representation of the class. The algorithm classifies data points based on their similarity, measured using Euclidean distance.
- Applying Feature Engineering techniques to extract data from URLs in order to implement the proposed algorithm for phishing URL detection

PART 3

THEORETICAL BACKGROUND

3.1. MACHINE LEARNING

Machine learning, situated within the field of artificial intelligence (AI) and computer science, is a discipline dedicated to leveraging data and algorithms to simulate human learning processes, aiming to progressively enhance accuracy. Figure 3.1 provides a visual representation of this concept [35].

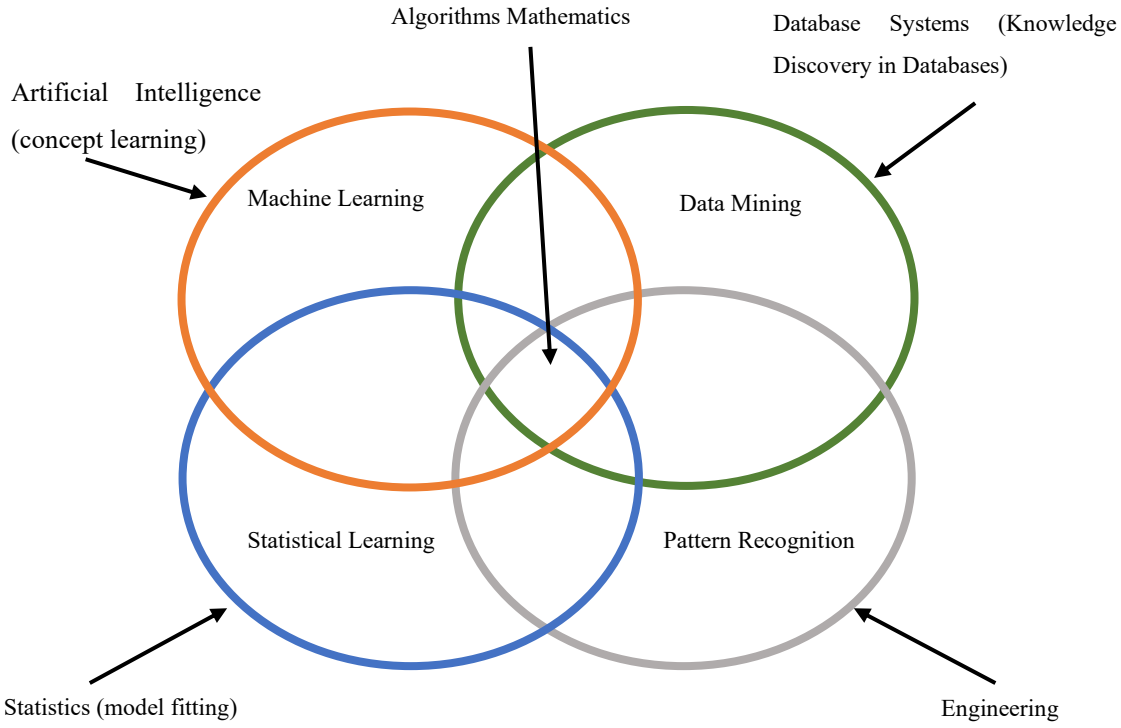

Figure 3.1. Malti Domain View [35].

Emulating human learning processes, such as those shown in Figure 3.2, is the means by which this goal can be accomplished. Machine learning algorithms have the capability to improve their analytical precision through a process called iterative

iterations. This allows the algorithms to autonomously update themselves with new insights using information gained from the analysis of data [36].

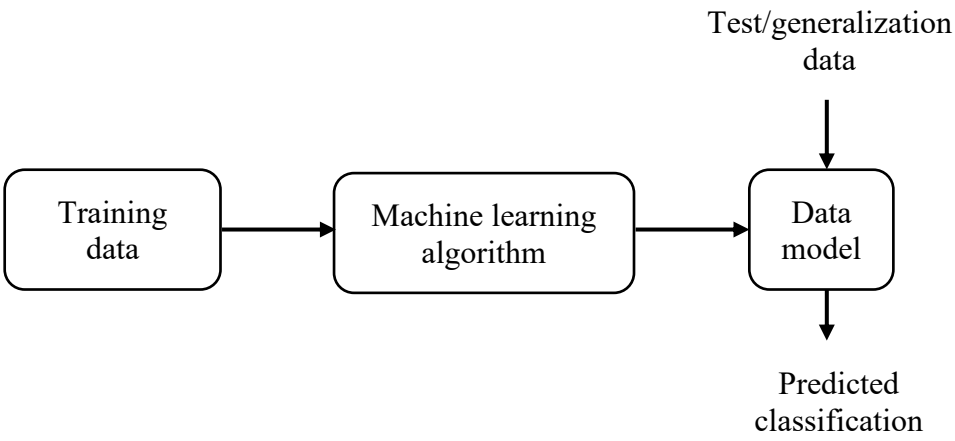

Figure 3.2. general schema for machine learning methods [35].

The iterative learning approach that is used in machine learning is distinguished by its one-of-a-kind nature and high value due to the fact that it enables algorithms to discover dormant insights without being given any explicit direction. This process of automatic learning distinguishes it from others because it makes the acquisition of new information easier. Applications of machine learning algorithms can be found in many different fields, including cybersecurity, medicine, email filtering, voice recognition, agriculture, and computer vision, all of which are areas in which the development of traditional algorithms to carry out these tasks would be difficult or even impossible [37, 38].

The rapid development of machine learning can be attributed to the increased accessibility of large data sets as well as the advancements in computing power that have been made in recent years. As a direct consequence of this, machine learning has emerged as an essential tool for addressing complex problems and improving decision-making processes across a wide variety of domains.

3.2. TYPES OF MACHINE LEARNING

The degree of human intervention in the raw data has an effect on the various types of machine learning models, and this influence can take the form of a variety of factors,

such as rewards, detailed feedback, or labels. There are many different approaches to machine learning, but one of the ones that is used the most frequently is:

3.2.1. Supervised Learning

Supervised learning is a category of machine learning that involves training a model on labeled data and subsequently utilizing this training to make predictions on new, unlabeled data. The primary objective of supervised learning is to forecast the target label for unseen data by leveraging the discovered patterns and relationships during the training phase [39]. Supervised learning encompasses a broad range of tasks, including image classification, speech recognition, and natural language processing. The algorithm takes a set of features as inputs and produces a set of labels as outputs. The objective is to discern the underlying relationship between the inputs and outputs. In addition to classification algorithms, supervised learning includes two other types: regression algorithms and classification algorithms. Regression algorithms are employed to predict continuous values, such as stock prices or tomorrow's weather, while classification algorithms are utilized to predict categorical values, such as animal types in an image or the sentiment of a text [40]. Supervised algorithms operate by identifying patterns in the relationship between the inputs (features) and outputs (labels) of the training data. The algorithm initially makes an initial estimation, which is iteratively refined and enhanced as it receives additional training data [41].
The overall process can be divided into the following steps:

- Acquisition and preparation of training data: This involves selecting appropriate features and labels and preprocessing the data to eliminate any noise or outliers.
- Selection and training of a suitable model: A well-suited model is chosen and trained using the prepared training data, taking into consideration the problem type, feature type, and data characteristics.
- Model evaluation: The trained model is assessed against a separate set of test data to evaluate its accuracy and identify areas for improvement, if necessary.
- Prediction on new data: Once the model has been trained and tested, it can be utilized to make predictions on unseen data.

The accuracy of the model, influenced by the quality of the training data and the chosen model, determines the quality of the predictions. Supervised algorithms aim to minimize prediction errors on the training data while avoiding overfitting, where the model becomes overly complex and fits the training data too closely, hindering its generalization to new data [42].

3.2.2. Unsupervised Learning

Unsupervised learning encompasses a machine learning approach in which models are trained on unlabeled data to uncover inherent patterns and relationships without a predefined prediction task. The main objective of unsupervised learning is to reveal underlying structures and detect latent patterns that may not be readily apparent through visual inspection alone [43]. Unsupervised learning algorithms operate without the need for labeled data and are employed for various tasks such as clustering, dimensionality reduction, and outlier detection. Clustering algorithms group similar data points based on defined similarity measures, such as grouping customers with similar spending behaviors [44]. Dimensionality reduction techniques aim to retain crucial information while reducing the number of data features, facilitating data visualization or further analysis by other machine learning algorithms. Anomaly detection algorithms identify data points that significantly deviate from the rest of the dataset, allowing the identification of outliers or anomalies, such as detecting fraudulent transactions [45, 46]. Unsupervised learning is particularly valuable when labeled data are scarce, costly to obtain, or when the objective is to uncover patterns without a specific prediction task in mind. Unsupervised algorithms autonomously discover patterns and structures within the data without prior knowledge of these patterns. By processing and analyzing the data, these algorithms establish connections between data points to uncover hidden patterns and structures. The specific method employed depends on the task and the algorithm used [47].

The unsupervised learning process can be summarized in the following steps:

- Data acquisition and preprocessing to remove noise and outliers.

- Selection of a suitable unsupervised learning algorithm based on the task and data characteristics.
- Training the chosen algorithm to identify patterns and relationships in the prepared data.
- Evaluation of the trained model's performance and potential adjustments.
- Interpretation of the model's results to extract meaningful information from the data.

The quality of the results obtained in unsupervised learning relies on the quality of the data and the chosen algorithm. Unsupervised algorithms strive to unveil the underlying data structure and discover meaningful patterns. However, evaluating the results of unsupervised learning algorithms can be challenging due to the absence of labeled data for comparison.

3.2.3. Semi-Supervised Learning

Semi-supervised learning is a form of machine learning that combines aspects of both supervised and unsupervised learning. In this approach, the model is trained on a combination of labeled and unlabeled data with the objective of leveraging the unlabeled data to enhance the model's performance on the labeled data [48-50].

Semi-supervised learning proves beneficial when obtaining labeled data is challenging or expensive, while a significant amount of unlabeled data is available. The idea is to utilize the unlabeled data to improve the model's understanding of the underlying data structure, thereby enhancing its performance on the labeled data [51]. Semi-supervised learning algorithms capitalize on the trade-off between the demand for a large labeled dataset in supervised learning and the abundance of unlabeled data. These algorithms employ the unlabeled data to make informed estimations about the labels and then refine these estimations using the labeled data [52]. In the process of semi-supervised learning, the steps resemble those of supervised learning, with the inclusion of incorporating the unlabeled data during the training phase. The quality of the results depends on the data quality and the choice of algorithm. Evaluation metrics such as

accuracy, precision, and recall are commonly employed to assess the performance of semi-supervised algorithms.

3.3. ALGORITHM

In the domain of machine learning, an algorithm refers to a set of instructions that guides a computer program in performing specific tasks, such as identifying patterns in data, generating predictions, or making decisions. Essentially, algorithms serve as the fundamental components of machine learning [53-55]. They represent a systematic approach for problem-solving or achieving specific objectives. In machine learning, algorithms are employed to construct models based on past data, enabling predictions to be made on new, unseen data.

The field of machine learning encompasses a diverse range of algorithms, including supervised learning algorithms, unsupervised learning algorithms, semi-supervised learning algorithms, among others. Each algorithm follows a distinct approach to the learning process, rendering them suitable for specific categories of challenges and data. The selection of an appropriate algorithm should be guided by the nature of the problem at hand and the characteristics of the data being utilized [55-58].

3.3.1. K-Nearest Neighbors Algorithm

The K-Nearest Neighbors (KNN) method is a supervised machine learning approach applicable to both classification and regression tasks [59, 60]. With the KNN algorithm, the objective is to predict the category or value of a given data point by identifying the neighboring points in the feature space that are closest to it [61, 62] Figure 3.3. illustrates the process visually.

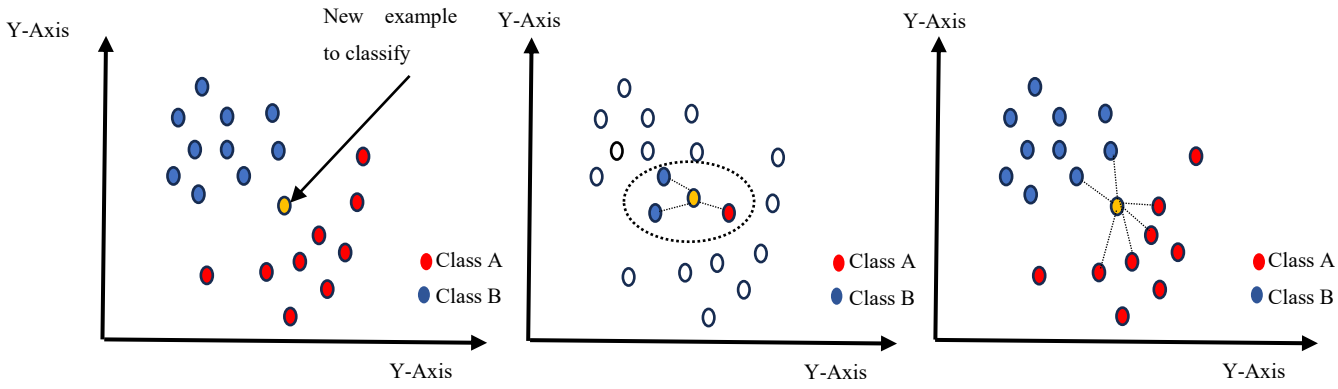

Figure 3.3. KNN Algorithm Diagram [62].

The K-Nearest Neighbors (KNN) method operates by storing all available data points and, for each new data point, identifying the K closest neighbors based on a distance metric such as Euclidean distance [63]. This process is repeated for all available data points. Subsequently, the category or value of the new data point is determined by majority voting among its K closest neighbors [64]. In classification, the category of a new data point is determined by the majority category among its K closest neighbors. For regression, the value of a new data point is computed by averaging the values of its K closest neighbors. KNN is relatively straightforward to implement and computationally efficient for small datasets. However, its computational complexity increases for large datasets. Hence, it may not be suitable for large-scale data analysis. Moreover, the choice of K is a crucial parameter that affects the precision of the method. Selecting a smaller K can make the algorithm more sensitive to outliers, leading to inaccurate predictions, while a larger K may result in difficulty distinguishing between distinct classes or values.

However, KNN is a versatile and robust algorithm applicable to various problem domains. However, its performance is contingent on the choice of distance metric and the value of K. Therefore, it is essential to carefully evaluate the algorithm's performance for the specific dataset and problem at hand [65].

3.3.1.1. Compute KNN

The implementation of the K-Nearest Neighbors (KNN) algorithm involves several steps:

- Data collection and preparation are performed, where data is gathered and preprocessed to eliminate noise and outliers. The dataset is then divided into training and testing sets, with the former used for model training and the latter for performance evaluation [66].
- The suitable distance metric is selected for the KNN algorithm, such as Euclidean distance, Manhattan distance, or Cosine similarity. The choice of distance metric depends on the specific task and dataset. Additionally, the value of K, representing the number of closest neighbors to consider, is determined. Typically, an odd value of K is chosen to avoid ties in majority voting.
- To evaluate the performance of the KNN model, its predictions on the testing set are compared to the actual class or value of the data points. Common evaluation metrics include accuracy, precision, recall, and F1 score [68]. If the model's performance is not satisfactory, adjustments can be made to the value of K, the distance metric, or the representation of data features to enhance its performance.

In summary, the computation of the KNN algorithm encompasses selecting a distance metric, determining the value of K, training the model on the training set, making predictions on new data points, and evaluating the model's performance.

3.3.2. Distance Matrix

A distance matrix refers to a square matrix that represents the pairwise distances between a collection of objects [67-69]. This matrix is commonly known as a distance matrix and serves as a two-dimensional table where each row and column correspond to a distinct item. Each element within the matrix indicates the distance separating the respective objects [70].

For example, in a distance matrix for a set of points in a two-dimensional space, each row and column represent a point, and the matrix elements signify the Euclidean distance between the points. In this case, the distance matrix is employed to measure the distances between the points [71]. The number of rows and columns in a distance matrix corresponds to the number of points within the dataset.

Distance matrices find extensive application in computer science and mathematics, particularly in domains such as machine learning, computer vision, and image processing. They are particularly useful in clustering algorithms, which aim to group comparable items based on their distances from one another. In such methods, the distance matrix is utilized to calculate the distances between items, which are subsequently used to estimate the similarity between objects. In other words, the distances between objects serve as a basis for determining their similarity [72, 73].

Various types of distance matrices are employed across different fields, including:

- Euclidean Distance Matrix: This commonly used distance matrix calculates the straight-line distance between two points in a multi-dimensional space using the Euclidean distance formula. It is represented by the formula:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.1}$$

- Manhattan Distance Matrix: Also known as taxicab distance, this matrix measures the sum of absolute differences between the coordinates of two points in a multi-dimensional space. It is represented by the formula:

$$d(x,y) = \left(\sum_{i=1}^{m}|x_i - y_i|\right) \tag{3.2}$$

- Minkowski Distance Matrix: A generalization of Euclidean and Manhattan distance matrices, the Minkowski distance matrix utilizes the Minkowski metric to calculate distances. It is represented by the formula:

$$\alpha(x,y) = \left(\sum_{l=1}^{n}\left|x_{i_1} - y_i\right|\right)^{\frac{1}{p}} \tag{3.3}$$

- Hamming Distance Matrix: This matrix determines the dissimilarity between two strings of the same length by counting the number of positions where their elements differ. It finds applications in error-correcting codes and encryption. It is represented by the formula:

$$D^H = \left(\sum_{i=1}^{k} |x_i - y_j| \right) \tag{3.4}$$

Each distance matrix has its own strengths and weaknesses, catering to different data types and problem domains. The selection of a distance matrix depends on the specific characteristics of the data and the objectives of the analysis.

In the context of the k-nearest neighbors (KNN) algorithm, a distance matrix is employed to store the distances between a set of test points and a set of training points. The objective of the KNN algorithm is to classify new points based on the classes of their k nearest neighbors in the training set.

To assign a new point to a category, the KNN algorithm calculates its distances from all points in the training set, which are stored in a distance matrix. Subsequently, the algorithm selects the k closest neighbors based on their distances from the center point. Finally, the new point is assigned to the class that shares the most similarities with its k nearest neighbors.

The choice of distance metric in the KNN algorithm significantly impacts its outcomes. For instance, the Euclidean distance is typically suitable for Euclidean data, while the Jaccard distance may be preferable for binary data. The selection of the distance metric should align with the data structure and the analysis goals.

In spaces with more than two dimensions, the Euclidean distance represents the straight-line distance between two points. It is named after the Greek mathematician Euclid and is computed as the square root of the sum of the squared differences between the coordinates of the two points. The Euclidean distance proves useful in various applications such as machine learning, data analysis, and computer vision. It particularly shines when the data is continuous and evenly distributed, providing an

effective measure of the distance between two points. Nevertheless, it is important to note that in certain scenarios, the Manhattan distance or cosine distance may serve as better measures of distance. The selection of the appropriate distance metric depends on the characteristics of the data and the specific insights sought in the analysis.

3.3.2. K-Mean Algorithm

K-Means is a widely recognized unsupervised machine learning method that clusters individual data points into distinct groups called clusters. The first algorithm of K-Means was invented by Stuart Lloys in 1957 [74, 75]. The primary objective of K-Means is to partition a dataset into K clusters, with each data point assigned to the cluster that has the closest mean. This is achieved by iteratively moving the cluster centroids closer to the mean of the data points assigned to each cluster until convergence is reached[76-79].

The K-Means method can be divided into the following steps:

- Initialize the centroids of the K clusters. This can be done randomly or using a heuristic technique.
- Assign each data point to the cluster centroid that is nearest to it. This involves calculating the Euclidean distance between each data point and the K centroids.
- Recalculate the mean of the data points assigned to each cluster and update the centroids accordingly.
- Repeat steps 2 and 3 until the centroids stop moving or the maximum number of iterations is reached.
- The final outcome is a clustering of the data points into K groups, with each data point placed in the cluster that is closest to it.

K-Means is commonly used for exploratory data analysis and is also employed as a preprocessing step for other machine learning algorithms. Despite being a fast and straightforward algorithm, K-Means is sensitive to the initial conditions and can become trapped in local minima. To address these limitations, alternative iterations of

the K-Means algorithm, such as K-Medoids and Fuzzy K-Means, have been developed[80, 81].

3.3.3. Enhancing Machine Learning Algorithms

The enhancement of machine learning algorithms is an ongoing process that can be achieved through various methods[82-86], including the following approaches:

- Feature Engineering: This process involves creating new features by combining or transforming existing features to improve the model's performance. By doing so, the model can learn more accurate correlations between the input and output variables.
- Hyperparameter Tuning: Machine learning algorithms have several hyperparameters that need to be configured before training. Fine-tuning these hyperparameters can significantly improve the overall performance of the model.
- Ensemble Methods: Utilizing an ensemble of multiple models, as opposed to a single model, can often lead to superior performance. Ensembles combine different models to capture various patterns in the data, resulting in more robust predictions.
- Transfer Learning: Reusing pre-trained models on similar tasks can be an effective approach to enhance performance. Pre-trained models have already learned relevant features from the data they were trained on, and this knowledge can be transferred to new tasks.
- Regularization: Incorporating regularization terms into the loss function can help reduce overfitting and improve the model's generalization ability.
- Data Augmentation: Generating additional samples from existing data can expand the dataset and assist the model in becoming more generalizable.
- Algorithm Selection: Choosing the appropriate algorithm for a specific task can significantly impact its performance. It is crucial to have a comprehensive understanding of the advantages and limitations of each algorithm to select the most suitable one for addressing the problem at hand.

These are just a few examples of the numerous approaches available for improving machine learning algorithms. Given the continuous discoveries in this field, it is crucial to continually explore different strategies and develop innovative techniques.

In machine learning, it is common to combine supervised and unsupervised learning algorithms to enhance the performance of the final model[21, 87, 88]. This is because each type of algorithm has its own strengths, which can be leveraged to improve overall performance. Supervised learning algorithms are trained on labeled data and make predictions based on the relationship between inputs and outputs. They are commonly used for classification and prediction tasks.

In contrast, unsupervised learning algorithms are trained on unlabeled data. They seek patterns and structures in the data without prior knowledge of the desired output. Unsupervised learning is often applied in clustering, dimensionality reduction, and outlier detection. By combining supervised and unsupervised algorithms, it is possible to take advantage of their respective strengths and improve performance. For instance, unsupervised algorithms can preprocess and extract features from the data, which can then serve as inputs for supervised learning algorithms. This enables the supervised algorithm to learn from a more abstract representation of the data, leading to better results.

Another approach involves leveraging a supervised learning algorithm to label the data generated by an unsupervised algorithm. This additional information can enhance the understanding of the data structure by the unsupervised algorithm. Combining supervised and unsupervised algorithms is also beneficial in semi-supervised learning scenarios, where only a limited amount of labeled data is available. Unsupervised algorithms can generate synthetic data that can be labeled and used to train a supervised algorithm [89, 90].

3.4. DOMAIN (PHISHING URLS)

Phishing refers to the deceptive practice of attempting to acquire sensitive information, such as usernames, passwords, and credit card numbers, by masquerading as a trustworthy entity through bulk emails. It aims to bypass spam filters and is also known as spear phishing and email phishing. Commonly, phishing involves sending

fraudulent emails to unsuspecting individuals, pretending to be well-known social networking sites, banks, auction platforms, or IT administrators. This type of social engineering relies on dishonesty to commit criminal activities [91]. The term "phishing" was first used by a renowned hacker and spammer in 1996 within the hacking program called AOHel [1].

The magnitude of phishing attacks has reached significant milestones, with APWG recording 1,097,811 total phishing incidents in the second quarter of 2022. The third quarter of the same year marked the highest recorded number of phishing attacks ever documented by APWG, reaching a total of 1,270,883 incidents. The peak month for attacks was August 2022, with 430,141 recorded incidents. The number of attacks has surged more than fivefold since the first quarter of 2020, when APWG reported 230,554 phishing incidents [92, 93].

The increase in attacks during Q3 2022 can be attributed to the targeting of specific entities, as persistent phishers made numerous attempts to compromise these targets [1]. Research conducted by OpSec Security, a founding member of APWG, revealed that phishing attacks in the financial sector (FS) continued to dominate, accounting for 23.2% of all phishing incidents in Q3 2022, down from 27.6% in Q2. The percentage of attacks on webmail and SAAS providers remained stable, while assaults on retail/ecommerce sites decreased to 4.1% from 14.6% in the first quarter. Phishing attempts targeting social media companies experienced a decline after ranging from 8.5% in 4Q2021 to 15.5% in 2Q2022. With the volatility of the crypto market and declining prices, phishing attempts against cryptocurrency targets, including cryptocurrency exchanges and wallet providers, decreased from 4.5% in Q2 to 2.0% in Q3.

Matthew Harris, Senior Product Manager, Fraud at OpSec Security, noted a significant increase in fraud volume within the Logistics and Shipping sector, particularly due to a surge in phishing attacks targeting the U.S. Postal Service. The detection levels of vishing (voice phishing) nearly tripled compared to Q2, continuing the trend observed in the second quarter [1].

3.4.1. Type of Phishing Attacks

Phishing attacks encompass a wide variety of deceptive techniques that are utilized by cybercriminals to trick individuals into divulging sensitive information or performing malicious actions. Phishing attacks come in a variety of forms, one of which is known as "credential phishing." Users are led to believe that they are interacting with a legitimate platform, such as a bank or a social media network, when in reality, the attacker is attempting to trick them into divulging their login information by sending them fraudulent emails or designing fraudulent websites. Another common form is known as "spear phishing," and it refers to attacks that are both personalized and targeted so that they are directed at particular people or businesses. These attacks frequently make use of information about the target that is already in the public domain in order to craft convincing messages or to impersonate trusted contacts. In addition, there is a technique known as "smishing," which is a form of phishing that is carried out through SMS text messages, and "vishing," which is a method of deceiving victims that uses voice communication channels, such as phone calls. In general, it is essential for individuals and organizations to have a solid understanding of the various types of phishing attacks in order to strengthen their cybersecurity posture and better protect themselves from these kinds of malicious endeavors [94, 95].

PART 4

METHODOLOGY

This study introduces a novel algorithm that draws inspiration from the K-Nearest Neighbors (KNN) approach, specifically designed for the identification of phishing URLs. The algorithm's primary objective is to enhance the current state-of-the-art in terms of efficiency, accuracy, and scalability. Its methodology involves determining the similarity between each point in a class and a unique point known as the core. The results obtained from rigorous testing demonstrate significant success, positioning the algorithm as a fitting solution for addressing the problem at hand. Comprised of several key steps, including Hyper algorithms, Hyperparameters, and iteration, these components work collaboratively to offer an efficient and effective solution to the problem. Extensive testing has been conducted on various test cases, substantiating its superiority over alternative approaches.

In the second stage of this study, an enhanced version of the algorithm called the Improve Core Classification Algorithm (ICCA) is introduced. ICCA accurately represents the class and arranges the points based on similarity votes. To leverage its potential, the study incorporates an active set (A_S) that calculates point distances using the Euclidean method. While it should be noted that the output of K-means algorithms may vary across implementations, this characteristic was exploited during the training model phase to improve overall accuracy.

This research thesis focuses specifically on Phishing URLs, which pertain to fraudulent websites aiming to deceive visitors into revealing personal information or credit card details. These URLs employ deceptive tactics to mislead users into believing they are engaging with a trustworthy source, while in reality, they are redirected to malicious content. The comprehensive analysis and detection of phishing URLs necessitate extensive preprocessing. Through these methods, raw URL

information undergoes transformation into a format suitable for utilization by machine learning algorithms and other analytical techniques.

4.1. DATA COLLECTION

Datasets have been sourced from various platforms, including Mendeley and Kaggle. Mendeley and Kaggle hold significant importance in the academic research and data science communities, respectively. Mendeley serves as a platform where researchers can share datasets with each other, primarily focusing on academic research. On the other hand, Kaggle is a platform that emphasizes data science competitions and facilitates access to diverse datasets, fostering collaboration among data scientists.

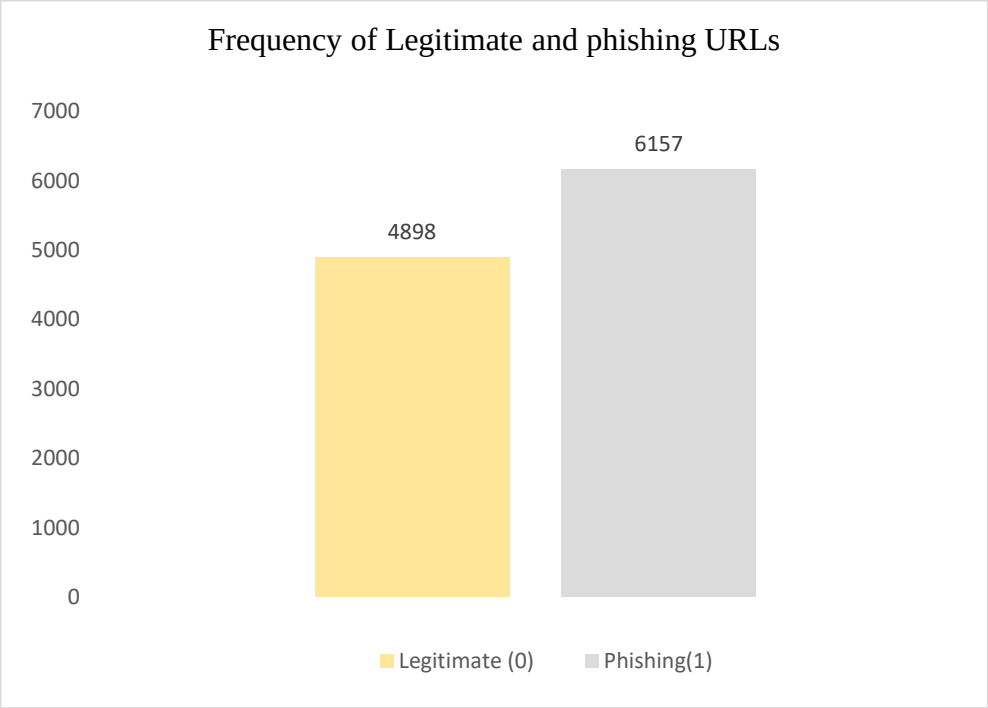

Figure 4.1. Phishing Websites Features [96].

The first dataset utilized in this study addresses the existing scarcity of high-quality training datasets. The researchers aim to fill this gap by identifying relevant features and expanding upon them. Their objective is to develop more comprehensive datasets that accurately capture the intricate nature of phishing sites. This endeavor will equip experts in the field with better tools to test hypotheses, refine algorithms, and enhance the accuracy of predicting phishing scams on websites. The first dataset consists of

11,056 instances and 31 features [96]. It encompasses both legitimate and phishing URLs, with 6,157 instances classified as phishing URLs and 4,898 instances classified as legitimate URLs (Figure 4.1). This dataset provides up-to-date and comprehensive information about phishing and legitimate websites, incorporating their distinctive features.

Hence, Moving on to the second dataset [97] It comprises 549,346 instances and is categorized into two groups, as illustrated in Figure 11. The first category is labeled as "Good," which signifies URLs that do not contain malicious content and are not classified as phishing sites. The second category is labeled as "Bad," representing URLs that contain malicious content and are classified as phishing sites.

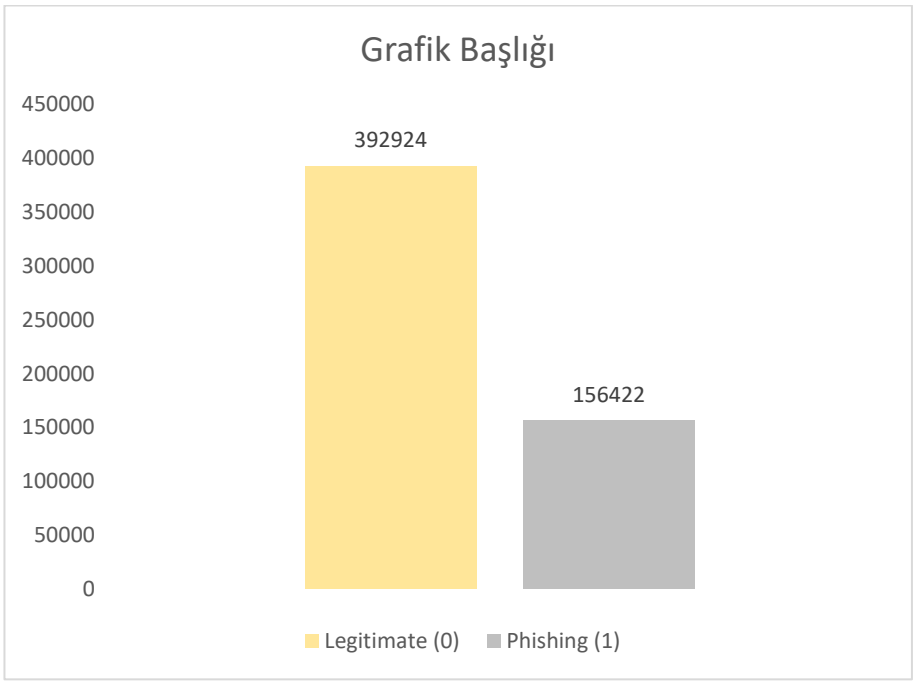

Figure 4.2. Phishing Site URLs [97].

In this thesis, a novel classification algorithm has been introduced. To provide a more comprehensive evaluation of its effectiveness and generalizability, the algorithm has been tested in a different domain. This approach aims to assess whether the algorithm's performance extends beyond its original domain of development.

However, by subjecting the algorithm to a different domain, we can examine its capacity to accurately classify suggestions and observe if its performance remains consistent across diverse types of data. Moreover, conducting tests in a different domain helps in identifying any inherent limitations or biases that the algorithm may possess. It enables us to investigate whether the algorithm is overfitting to the original domain or exhibiting domain-specific patterns that may not be applicable elsewhere. By evaluating the algorithm's performance in a new domain, we can determine its robustness and identify any necessary adjustments or adaptations required to enhance its accuracy and generalizability. One of the domains considered in this study is the Heart Disease dataset [98], comprising 3,656 instances and 15 features. Another domain is the Indian Liver Patient Record [99], which consists of 2,000 instances and 11 features. Lastly, the Cardiovascular Disease Dataset [100] includes 3,656 instances and 15 features. These datasets have been selected to assess the algorithm's performance and explore its applicability in different contexts.

4.2. CONFUSION MATRIX

A confusion matrix is a widely employed table for evaluating the performance of machine learning algorithms in classification tasks. It allows a comparison between the algorithm's predicted and actual classifications, revealing the number of instances that were correctly or incorrectly classified [101, 102]. Typically, a confusion matrix consists of four cells arranged in a 2x2 table. The rows represent the actual classes, while the columns represent the predicted classes. Each cell in the matrix represents the count of observations corresponding to a particular combination of predicted and actual classes.

The four cells of the confusion matrix are defined as follows:

True positive (TP): The number of instances correctly classified as positive by the algorithm.
False positive (FP): The number of instances incorrectly classified as positive by the algorithm.

True negative (TN): The number of instances correctly classified as negative by the algorithm.
False negative (FN): The number of instances incorrectly classified as negative by the algorithm.

Hence, by utilizing the confusion matrix, various performance metrics can be calculated, including accuracy, precision, recall, and F1 score. These metrics offer insights into the strengths and weaknesses of the machine learning algorithm. Analyzing the confusion matrix enables machine learning practitioners to gain a better understanding of the algorithm's error patterns and take necessary steps to enhance its performance.

4.2.1. Accuracy

Common performance metric that can be calculated from a confusion matrix in classification tasks. It measures the proportion of instances that were correctly classified by the machine learning algorithm, out of all the instances that were classified [102, 103].

The accuracy can be calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

where TP represents the number of true positives, TN represents the number of true negatives, FP represents the number of false positives, and FN represents the number of false negatives.

In other words, accuracy measures the overall correctness of the algorithm's classifications, regardless of the specific class. It provides a general idea of how well the algorithm is performing, but it may not be a suitable metric for imbalanced datasets where the distribution of classes is uneven. For example, if a machine learning

algorithm correctly classifies 90 out of 100 instances, its accuracy would be 90%. However, if the dataset has a class distribution of 90% positive and 10% negative, the algorithm may be classifying all instances as positive, resulting in high accuracy but poor performance on the negative class. In such cases, precision, recall, or F1 score may be more appropriate metrics to evaluate the algorithm's performance.

4.2. PRECISION

Precision is an essential performance metric derived from a confusion matrix in classification tasks. It quantifies the proportion of true positive classifications made by a machine learning algorithm out of all positive classifications made by the same algorithm [104].

The calculation of precision involves the use of the following formula:

$$Precision = TP/(TP + FP) \tag{4.2}$$

Here, TP represents the count of true positives, and FP represents the count of false positives.

Essentially, precision evaluates the accuracy of positive predictions generated by the algorithm. It becomes particularly valuable when the cost associated with false positives is high, such as in medical diagnosis or fraud detection scenarios.

4.3. RECALL

The concept of recall, also known as sensitivity or true positive rate, holds great significance in the domains of machine learning and data analysis. It refers to the ability of a classification model to correctly identify all positive instances within a given dataset. Recall serves as a measure of a model's capacity in this regard.

In the context of binary classification problems, recall is defined as the ratio of true positives to the total number of actual positives. It quantifies the model's ability to capture all relevant positive occurrences.

In situations where the consequences of false negatives are significant, such as in medical diagnosis or security screening, recall becomes a crucial metric to consider. It is also valuable in general as it indicates the model's effectiveness in minimizing the occurrence of false negatives and, consequently, reducing the risk of overlooking important events [105].

The calculation of recall can be performed using the following formula:

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

Here, TP represents the count of true positives, and FN represents the count of false negatives.

4.4. F1 SCORE

The F1 score, commonly employed in the fields of machine learning and data analysis, serves as a performance measure that combines the metrics of accuracy and recall into a single value. It represents the harmonic mean of accuracy and recall, thereby assigning equal importance to both measures. The F1 score is computed by taking the weighted average of precision and recall scores, with 1.0 indicating the best possible score and 0.0 representing the worst [105]. The F1 score is particularly valuable in binary classification problems where there is an imbalance between positive and negative cases. In such cases, a classifier that consistently predicts negative examples may achieve high accuracy but would demonstrate poor precision and recall. The F1 score offers a means to evaluate the overall performance of the classifier by considering both its accuracy and recall.

The formula for calculating the F1 score is as follows:

$$F_1 Score = 2\frac{Precision * recall}{presion + recall} \quad (4.4)$$

In this formula, precision and recall represent the respective performance metrics. The F1 score provides a comprehensive assessment of the classifier's effectiveness in handling imbalanced classification scenarios, accounting for both accuracy and recall.

4.3. PREPROCESSING

The preprocessing of phishing URLs serves as a foundation for developing robust cybersecurity systems. By analyzing and extracting relevant features from URLs, security professionals can proactively identify potential phishing attempts, effectively protecting individuals and organizations from falling victim to fraudulent schemes. The application of advanced analysis techniques, such as feature extraction and selection, allows for the identification of suspicious patterns and anomalies in URLs that may indicate malicious intent. Additionally, the integration of machine learning algorithms enables the development of sophisticated models that consider contextual information, semantic structures [106, 107] and behavioral patterns, significantly improving the accuracy and effectiveness of phishing detection systems. However, the preprocessing of phishing URLs is a fundamental step in enhancing cybersecurity measures, enabling early detection and prevention of phishing attacks in today's evolving threat landscape [108].

In this thesis, we implement the preprocessing for the "Phishing site URLs" dataset, which is the official name[97]. The dataset currently contains 549,346 entries in two columns. The prediction column consists of two types of labels:

A. Good: This indicates that the site is not a phishing site, and the URLs do not contain malicious content.

B. Bad: This indicates that the site is a phishing site, and its URLs contain malicious content.

Using Jupyter Notebook and the Python language, we performed the preprocessing steps on the URLs as follows:

In our thesis, we have selected a specific definition for the parts of the URL, acknowledging the existence of various definitions in the literature. However, for the purpose of our research, we have adopted the following definition:

URL: https://www.example.com/Path/to/resource?param1=value1¶m2=value2#section1

- Domain: www.example.com
- Path: /path/to/resource
- Query: param1=value1¶m2=value2
- Fragment: section

In our work, we conducted an extensive analysis of URLs, focusing on various components such as the domain, path, query, and fragment. Additionally, we explored the significance of specific characters within URLs, including (- = ! + $. @ ~ * % ? & , # space), during the extraction process. Let's delve into the various components of a URL and gain a better understanding of their differences. By examining each part individually, we can grasp their distinct purposes and functions:

1- In our research, we focused on studying URLs and extracting important characters from them. We aimed to gain insights by analyzing various characteristics, including the length of the URL (url_length) and the quantities of specific characters present. Firstly, we examined the occurrence of periods (.) in the URL (qty_dot_url). These dots are significant as they often separate domain and subdomain names within the URL. Additionally, we analyzed the quantity of hyphens (-) in the URL (qty_hyphen_url). Hyphens can serve different purposes, such as improving readability or distinguishing between words in the domain. Furthermore, we investigated the presence of forward slashes (/) in the URL (qty_slash_url). These slashes indicate directory structures or parameters within the URL. Moreover, we counted the occurrence of question marks (?) in the URL (qty_questionmark_url). Question marks typically signify the start of query strings in URLs. Furthermore, we examined the quantity of equal signs (=) in the URL (qty_equal_url). Equal signs are commonly used in

URL parameters to assign values to specific variables. Additionally, we analyzed the presence of at symbols (@) in the URL (qty_at_url). While at symbols are not commonly found in domain names, they may have specific implications depending on their position within the URL. Furthermore, we looked for ampersands (&) in the URL (qty_and_url). Ampersands are often used as separators between different parameters in URL query strings. In addition, we investigated the occurrence of exclamation marks (!) in the URL (qty_exclamation_url). Exclamation marks can occasionally be used for emphasis or to indicate specific actions within URLs. Moreover, we examined the presence of spaces ( ) in the URL (qty_space_url). Although spaces are not valid characters in URLs, they may be encoded as %20 or replaced with other characters. Furthermore, we analyzed the quantity of tildes (~) in the URL (qty_tilde_url). Tildes can be used for various purposes, such as indicating user directories or serving as placeholders in URL patterns. Additionally, we looked for commas (,) in the URL (qty_comma_url). While commas are rarely used in URLs, they may hold specific meanings in certain contexts. Moreover, we investigated the occurrence of plus signs (+) in the URL (qty_plus_url). Plus, signs can sometimes replace spaces in URLs or indicate concatenation operations. Furthermore, we examined the presence of asterisks (*) in the URL (qty_asterisk_url). Asterisks may have special significance in wildcard patterns or act as placeholders in URL patterns. Additionally, we analyzed the quantity of hashtags (#) in the URL (qty_hashtag_url). Hashtags are typically associated with anchor links within a webpage and may not appear frequently in domain names. Moreover, we looked for dollar signs ($) in the URL (qty_dollar_url). While dollar signs are not commonly found in domain names, they may be used in specific URL contexts, such as indicating dynamic content. Lastly, we analyzed the quantity of percent signs (%) in the URL (qty_percent_url). Percent signs are often used in URL encoding to represent special characters or spaces. By considering these various characteristics and their quantities, we aimed to gain a deeper understanding of the structure and composition of URLs. This analysis provides valuable insights into the URL's components, which can have implications for security, SEO optimization, and overall website usability.

As we can see in Figure 4.3, the results of the hyphen distribution of the URL indicate a variance in distribution, with a higher occurrence in legitimate URLs compared to phishing URLs.

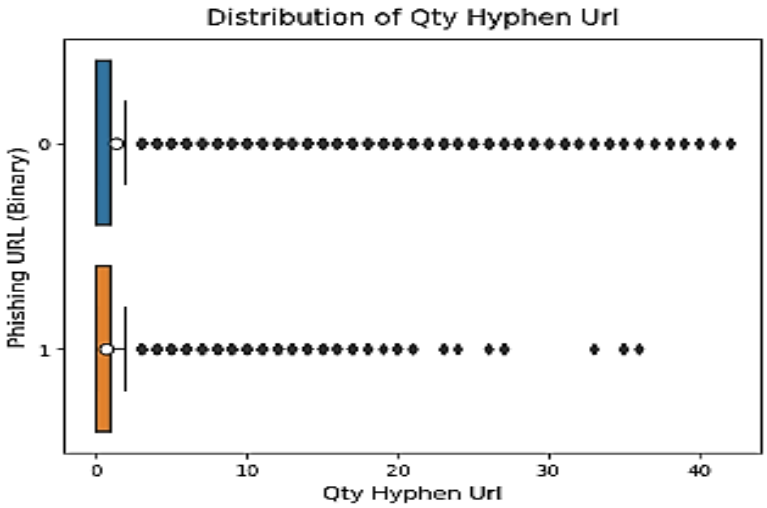

Figure 4.3. The DEA resut for hyphen distribution of the url.

As we can see in Figure 4.4, the results of the dot distribution of the URL indicate a variance in distribution, with a higher occurrence in phishing URLs compared to legitimate URLs.

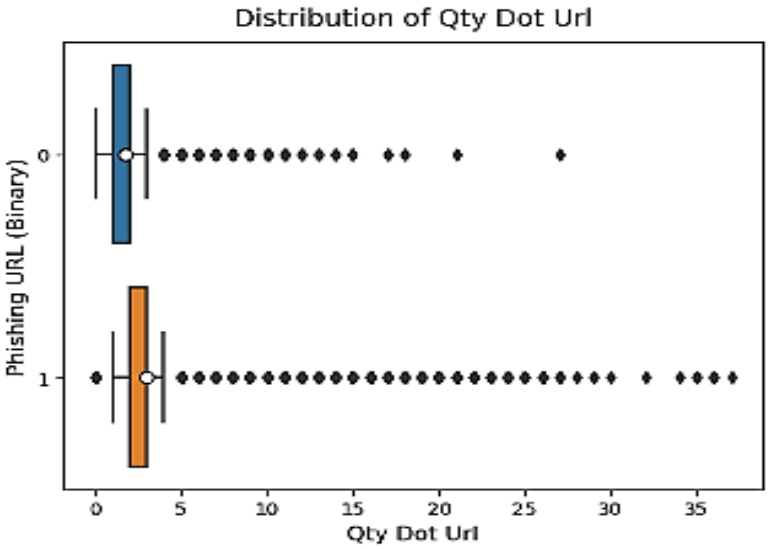

Figure 4.4. The DEA resut for dot distribution of the url.

As we can see in Figure 4.5, the results of the slash distribution of the URL indicate a variance in distribution, with a higher occurrence in phishing URLs compared to legitimate URLs.

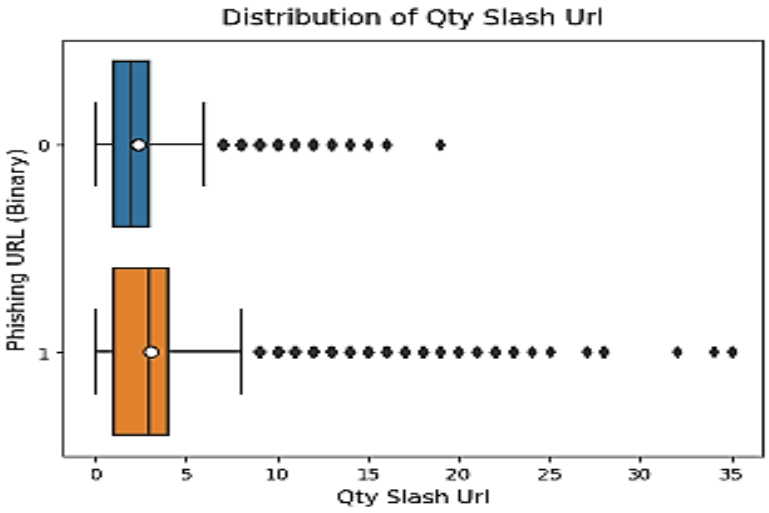

Figure 4.5. The DEA resut for slash distribution of the url.

By comparing the DEA results, we can conclude the importance of studying the characteristics of the URL.

2- Domain: The domain in a URL refers to the registered name that uniquely identifies a website on the internet. It is a fundamental part of the URL structure and is typically found after the protocol (e.g., "http://" or "https://") and before any additional path, query, or fragment components. The domain provides a human-readable and memorable name that serves as an entry point for accessing web resources. In simple terms, the domain represents the website's address or location on the internet. It consists of two main parts: the domain name and the top-level domain (TLD). The domain name is the specific identifier chosen by the website owner, which can be a combination of alphanumeric characters and hyphens. The TLD, on the other hand, is the extension that follows the domain name and indicates the purpose or type of organization associated with the website (e.g., ".com," ".org," ".edu," etc.). For example, in the URL "https://www.example.com/home," the domain is "www.example.com." Here, "example" is the domain name, and ".com" is the TLD. The domain plays a

crucial role in verifying the legitimacy and authenticity of a website. It serves as a recognizable label for users and helps establish trust when accessing online resources. Additionally, the domain can be analyzed and compared against known lists of malicious or phishing domains to identify potential threats or fraudulent activities. Understanding the domain within a URL is vital in various aspects of web security, such as phishing detection, website reputation analysis, and ensuring secure communication between users and websites. By examining the domain, security professionals can assess the credibility of a website and make informed decisions regarding its trustworthiness and potential risks associated with accessing its content.

In our research, we focused on studying domains and extracting specific characters from them. By analyzing these characters, we gained insights into various important characteristics of the domain. Here is an extended and proofread version of the text:

In our study, we conducted an in-depth analysis of domains and examined key characters within them. By scrutinizing these characters, we obtained valuable information about the domain's composition and structure. The following are some of the important characters we considered:

- Firstly, we evaluated the length of the domain (domain_length), which provided insights into the overall complexity and potential manipulation of the domain.
- Additionally, we counted the occurrences of periods (.) in the domain (qty_dot_domain). Periods play a crucial role in separating domain and subdomain names.
- Furthermore, we analyzed the quantity of hyphens (-) in the domain (qty_hyphen_domain). Hyphens are often utilized to enhance readability or differentiate words within the domain.
- Moreover, we investigated the presence of forward slashes (/) in the domain (qty_slash_domain). These slashes can indicate directory structures or subdirectories within the domain.

- We also examined the occurrence of question marks (?) in the domain (qty_questionmark_domain). Question marks commonly denote query strings within the domain.
- Additionally, we counted the quantity of equal signs (=) in the domain (qty_equal_domain). Equal signs are frequently used to assign values to variables within URL parameters.
- Furthermore, we analyzed the presence of at symbols (@) in the domain (qty_at_domain). While at symbols are not typically found in domain names, they may have specific implications depending on their usage.
- Moreover, we looked for ampersands (&) in the domain (qty_and_domain). Ampersands often serve as separators between different parameters within the domain.
- In addition, we investigated the occurrence of exclamation marks (!) in the domain (qty_exclamation_domain). Exclamation marks can be used for emphasis or to indicate specific actions within the domain.
- Furthermore, we examined the presence of spaces ( ) in the domain (qty_space_domain). While spaces are not valid characters in domain names, they may be encoded or substituted with other characters.
- Additionally, we analyzed the quantity of tildes (~) in the domain (qty_tilde_domain). Tildes can serve various purposes, such as indicating user directories or acting as placeholders in URL patterns.
- Moreover, we looked for commas (,) in the domain (qty_comma_domain). Although commas are not commonly used in domain names, they may hold specific meanings in certain contexts.
- Furthermore, we investigated the occurrence of plus signs (+) in the domain (qty_plus_domain). Plus, signs can replace spaces in some cases or indicate concatenation operations within the domain.
- Additionally, we examined the presence of asterisks (*) in the domain (qty_asterisk_domain). Asterisks may have special significance in wildcard patterns or act as placeholders within the domain.
- Furthermore, we analyzed the quantity of hashtags (#) in the domain (qty_hashtag_domain). Hashtags are typically associated with anchor links within webpages and may appear infrequently in domain names

- Moreover, we looked for dollar signs ($) in the domain (qty_dollar_domain). While dollar signs are not commonly found in domain names, they may be used in specific contexts, such as indicating dynamic content.
- Lastly, we analyzed the quantity of percent signs (%) in the domain (qty_percent_domain). Percent signs are often used in URL encoding to represent special characters or spaces.

By examining these various characters and their quantities, we gained a comprehensive understanding of the domain's composition and structure. This analysis provided valuable insights into the domain's characteristics, which can have implications for security, SEO optimization, and overall website usability.

As we can see in Figure 4.6, the results of the length distribution of the domain indicate a variance in distribution, with a higher occurrence in phishing URLs compared to legitimate URLs.

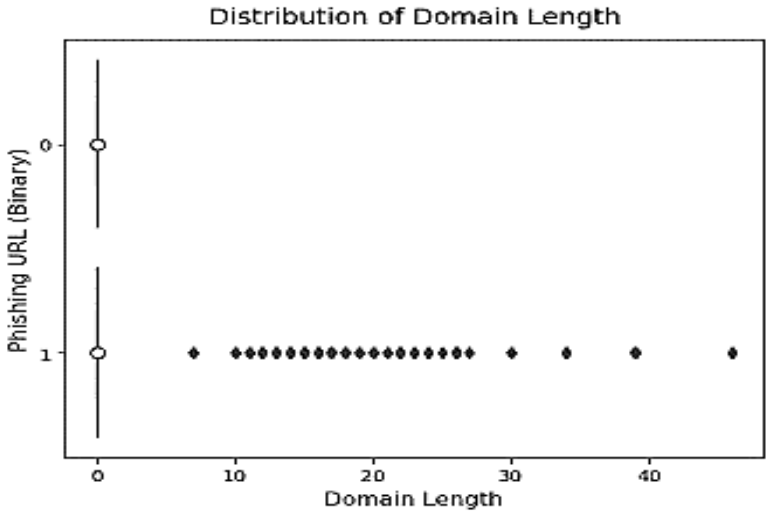

Figure 4.6. The DEA resut for length distribution of the domain.

As shown in Figure 4.7, the results of the domain's dot distribution imply a variance in distribution, with phishing URLs having a higher occurrence than legitimate URLs.

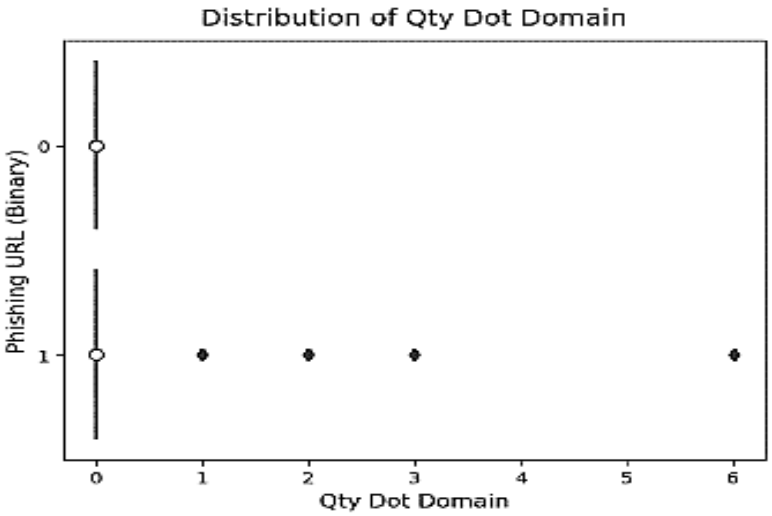

Figure 4.7. The DEA resut for dot distribution of the domain.

3- The path in a URL refers to the specific location or resource within a website's hierarchy. It follows the domain component and is separated by slashes ("/"). The path provides a way to navigate to a particular webpage or access a specific file or directory within a website. The path component of a URL often represents the organization and structure of a website's content. It can include directories, subdirectories, and file names. Each segment of the path is separated by slashes, indicating the hierarchical relationship between the elements.

For example, in the URL "https://www.example.com/path/to/resource", the path component is "/path/to/resource". Here, "path/to/resource" represents the specific location within the website's structure where the desired resource or webpage can be found. The path component of a URL is essential for resolving the correct webpage or resource on the server. It helps in organizing and categorizing content within a website, allowing users and applications to access specific information efficiently. Understanding the path component in a URL is important for analyzing website structures, identifying specific pages or resources, and resolving relative links within a website. It is a valuable element in web development, content management, and cybersecurity analysis.

In our work, we studied the path and extracted these characters from the path. We counted some of the important characters, such as the length of the path (path_length),

the quantity of dots in the path (qty_dot_path), the quantity of hyphens in the path (qty_hyphen_path), the quantity of slashes in the path (qty_slash_path), the quantity of question marks in the path (qty_questionmark_path), the quantity of equal signs in the path (qty_equal_path), the quantity of at symbols in the path (qty_at_path), the quantity of ampersands in the path (qty_and_path), the quantity of exclamation marks in the path (qty_exclamation_path), the quantity of spaces in the path (qty_space_path), the quantity of tildes in the path (qty_tilde_path), the quantity of commas in the path (qty_comma_path), the quantity of plus signs in the path (qty_plus_path), the quantity of asterisks in the path (qty_asterisk_path), the quantity of hashtags in the path (qty_hashtag_path), the quantity of dollar signs in the path (qty_dollar_path), and the quantity of percent signs in the path (qty_percent_path).

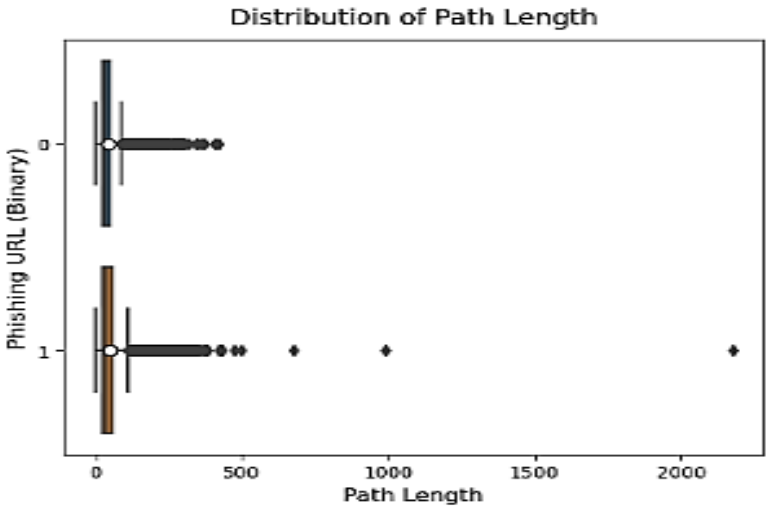

Figure 4.8. The DEA resut for length distribution of the path.

The findings of the length distribution of the path reveal a variance in distribution, as shown in Figure 4.8, with a higher occurrence in phishing URLs compared to genuine URLs. This difference can be attributed to the fact that phishing URLs are more likely to contain malicious content.

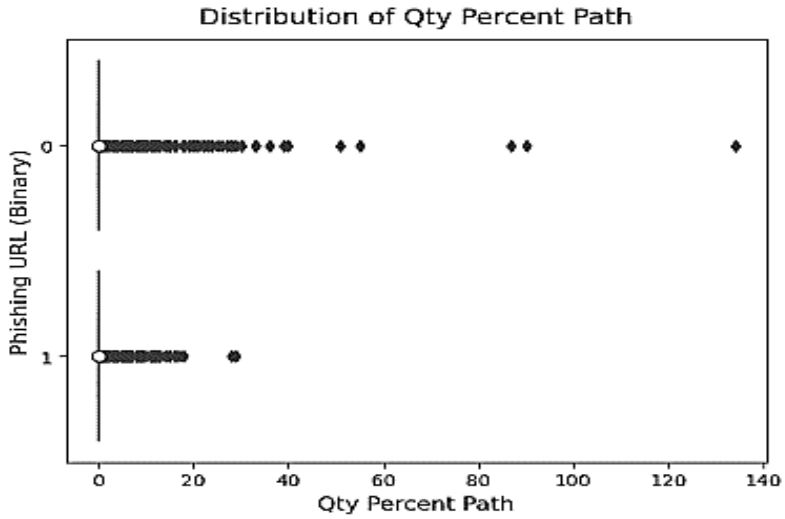

Figure 4.9. The DEA resut for percent distribution of the path.

As can be seen in Figure 4.9, the results of the percent distribution of the path suggest that there is a variance in distribution, with a higher occurrence in valid URLs compared to phishing URLs. This is because authentic URLs are more likely to contain the path than phishing URLs.

The findings depicted in Figure 4.9 illustrate that the slash distribution of the path exhibits a distribution variance, with a greater frequency of occurrence in authentic URLs in contrast to phishing URLs.

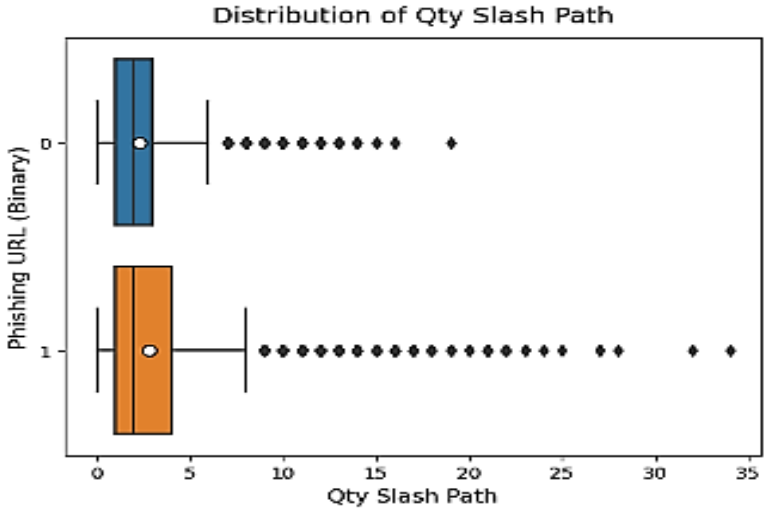

Figure 4. 10. The DEA resut for slash distribution of the path.

4- The query component in a URL is used to pass additional parameters or data to the server. It follows the path component and is preceded by a question mark ("?"). The query component consists of key-value pairs separated by ampersands ("&"), where each pair represents a specific parameter and its corresponding value. The query component is often employed in dynamic web pages or applications to transmit data from the client to the server. It allows for customization and interaction with web resources based on specific parameters. The data passed through the query component can be used for various purposes, such as filtering search results, specifying user preferences, or submitting form data. For example, in the URL "https://www.example.com/search?q=keyword&page=1", the query component is "?q=keyword&page=1". Here, "q" and "page" are the parameters, and "keyword" and "1" are the respective values associated with those parameters. The query component is flexible and can contain multiple parameters, each providing additional information to the server. The order of the parameters within the query component is generally arbitrary and does not affect the functionality of the URL.

In our work, we studied the query and extracted these characters from it. We counted some of the important characters, such as the length of the query (query_length), the quantity of dots in the query (qty_dot_query), the quantity of hyphens in the query (qty_hyphen_query), the quantity of slashes in the query (qty_slash_query), the quantity of at symbols in the query (qty_at_query), the quantity of ampersands in the query (qty_and_query), the quantity of exclamation marks in the query (qty_exclamation_query), the quantity of spaces in the query (qty_space_query), the quantity of tildes in the query (qty_tilde_query), the quantity of commas in the query (qty_comma_query), the quantity of plus signs in the query (qty_plus_query), the quantity of asterisks in the query (qty_asterisk_query), the quantity of hashtags in the query (qty_hashtag_query), the quantity of dollar signs in the query (qty_dollar_query), and the quantity of percent signs in the query (qty_percent_query).

Figure 4.11 displays the query length distribution results, which show a non-normal distribution with more phishing URLs having longer lengths than legitimate URLs.

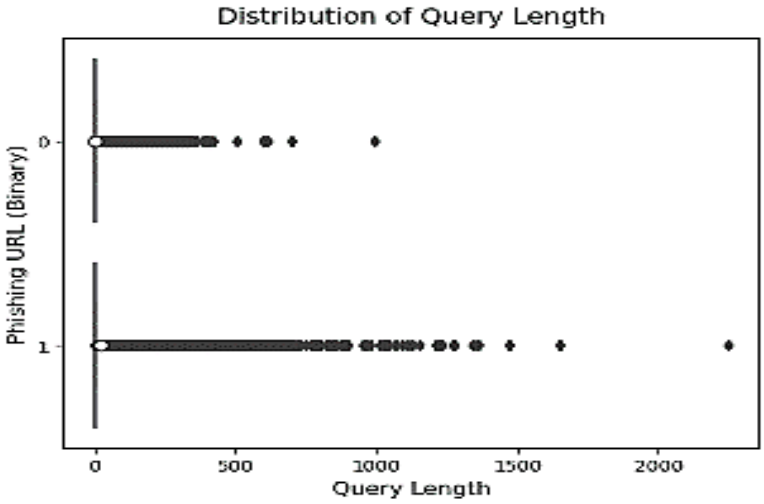

Figure 4.11. The DEA resut for lenght distribution of the query.

The results of the percent distribution of the query suggest that there is a variance in distribution, with a higher prevalence in phishing URLs compared to legal URLs. This can be seen in Figure 4.12, which presents these findings.

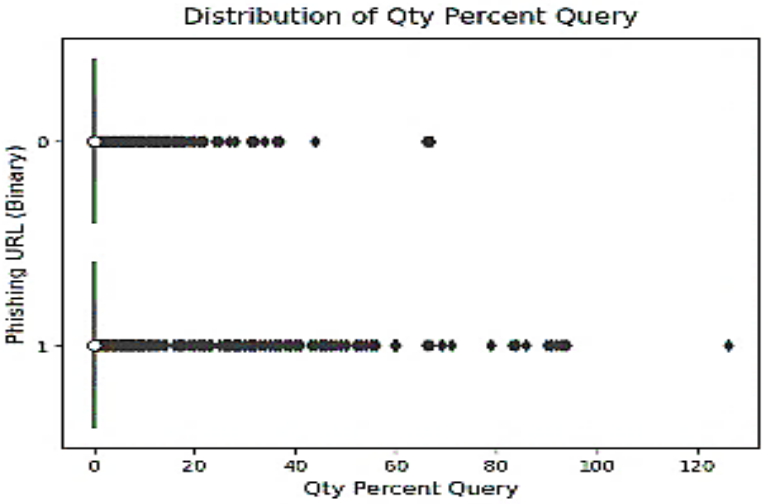

Figure 4.12. The DEA resut for percent distribution of the query.

The results of the hyphen distribution of the query suggest that there is a variance in distribution, with a higher prevalence in legal URLs compared to phishing URLs. This can be seen in Figure 4.13, which presents these findings.

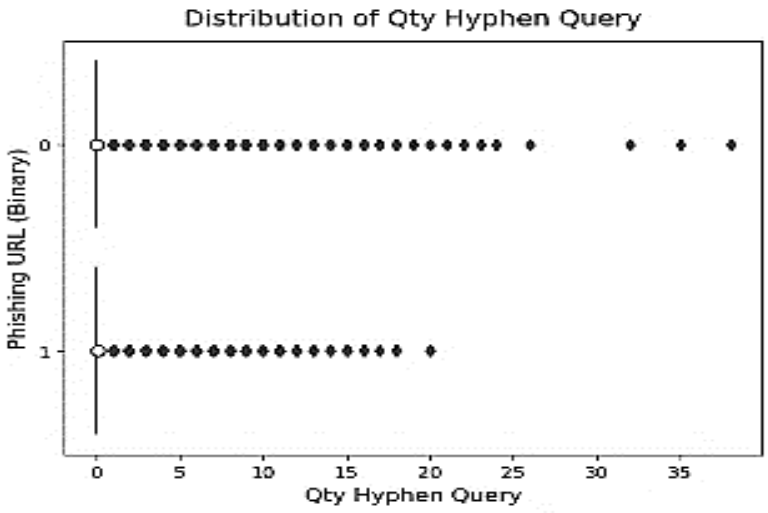

Figure 4. 13. The DEA resut for hyphen distribution of the query.

5- The fragment in a URL, also known as the URL fragment identifier or anchor, refers to a specific section or location within a webpage. It is denoted by a hash symbol ("#") followed by a fragment identifier. The fragment component is primarily used to navigate to a specific section of a webpage, allowing users to jump directly to a particular portion of the content. Web designers often use fragments to divide webpages into meaningful sections and provide quick access to specific information or sections of interest.

For example, in the URL "https://www.example.com/page#section1", the fragment component is "#section1". Here, "section1" refers to a designated section within the webpage that the URL points to. The fragment component is typically processed on the client-side, as it does not get transmitted to the server. Web browsers interpret the fragment and scroll the webpage to the corresponding section or anchor point identified by the fragment identifier.

The fragment component is commonly used in conjunction with HTML elements such as headings, paragraphs, or named anchors, where these elements are assigned specific IDs to serve as the targets for the fragment identifier. While the fragment component primarily affects the user's browsing experience, it can also be utilized for various purposes, such as bookmarking specific sections of a webpage or sharing a direct link to specific content within a page.

In our work, we extensively studied the fragment and meticulously extracted various characters from it. We conducted a comprehensive analysis, taking into account a range of important characters such as the length of the fragment (fragment_length), the quantity of periods (.) in the fragment (qty_dot_fragment), the quantity of hyphens (-) in the fragment (qty_hyphen_fragment), the quantity of slashes (/) in the fragment (qty_slash_fragment), the quantity of question marks (?) in the fragment (qty_questionmark_fragment), the quantity of equal signs (=) in the fragment (qty_equal_fragment), the quantity of at symbols (@) in the fragment (qty_at_fragment), the quantity of ampersands (&) in the fragment (qty_and_fragment), the quantity of exclamation marks (!) in the fragment (qty_exclamation_fragment), the quantity of spaces ( ) in the fragment (qty_space_fragment), the quantity of tildes (~) in the fragment (qty_tilde_fragment), the quantity of commas (,) in the fragment (qty_comma_fragment), the quantity of plus signs (+) in the fragment (qty_plus_fragment), the quantity of asterisks (*) in the fragment (qty_asterisk_fragment), the quantity of hashtags (#) in the fragment (qty_hashtag_fragment), the quantity of dollar signs ($) in the fragment (qty_dollar_fragment), and the quantity of percent signs (%) in the fragment (qty_percent_fragment).

By meticulously analyzing these characters, we gain valuable insights into the properties and composition of the fragment. This analysis allows us to make informed decisions and draw meaningful conclusions based on the specific characteristics of the fragment's content.

Phishing attacks threaten individuals and organizations, making detection and prevention crucial. Identifying key differences between legitimate and malicious URLs is essential to solving this problem. Traditional feature extraction methods use manual selection or heuristics, which are subjective and ineffective. Data Envelopment Analysis (DEA) overcomes these limitations with a systematic, data-driven approach. DEA can extract many features from phishing URLs. DEA can measure decision-making unit efficiency based on inputs and outputs. Phishing URLs are decision-making units, while their inputs and outputs are their attributes and characteristics. DEA identifies the most important and discriminating features that classify URLs as

legitimate or malicious. DEA's quantitative feature importance assessment is a major benefit of feature extraction. This method helps identify phishing URLs. DEA can handle high-dimensional datasets and noise and outliers in phishing detection. Its robustness makes the selected features reliable and effective at identifying phishing URLs.

DEA is used to optimize phishing URL feature extraction. DEA maximizes URL efficiency by considering its attributes, identifying the most efficient feature combination, and highlighting the most important classification factors. We can improve phishing URL classifiers by using DEA's strengths. The extracted features can improve anti-phishing systems.

DEA-based feature extraction can also help develop phishing attack mitigation strategies. Figures 4.3-4.7 show the results of our DEA-based dataset feature extraction of Domain, Path, Query, and Fragment. These figures illuminate how these features affect URL classification. DEA and feature extraction can significantly improve phishing URL detection. This research improves security systems, protecting users from phishing attacks. Our findings also encourage feature extraction research to address cybersecurity issues.

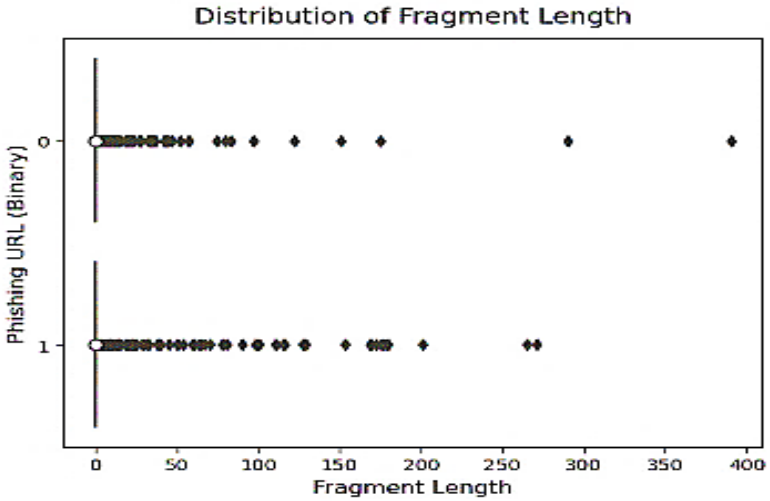

Figure 4.14. The DEA resut for length distribution of the fragment.

The findings of the length distribution of the fragment suggest a variance in distribution, with a slightly larger incidence in valid URLs compared to phishing URLs, as can be seen in Figure 4.14. This difference in occurrence is due to the fact that legitimate URLs are more likely to contain the fragment.

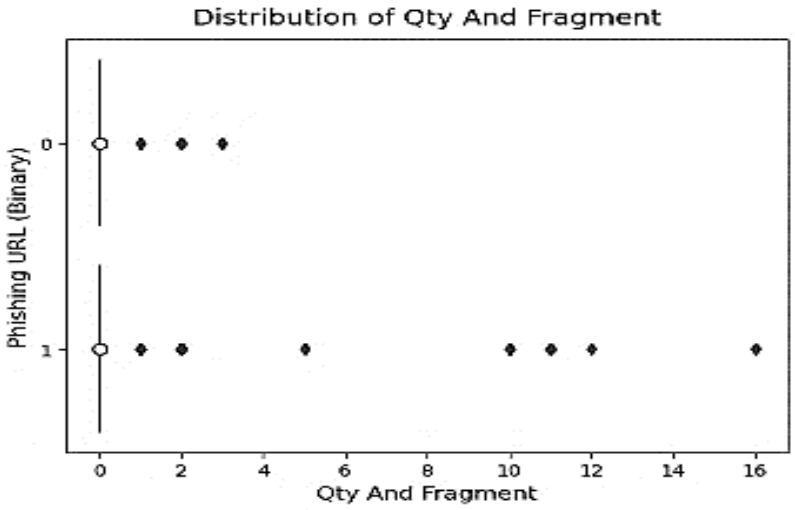

Figure 4.15. The DEA resut for and distribution of the fragment.

As can be seen in Figure 4.15, the results of the and distribution of the fragment suggest a variance in distribution, with a little larger occurrence in phishing URLs compared to valid URLs. This is because phishing URLs are more likely to contain the fragment than legitimate URLs.

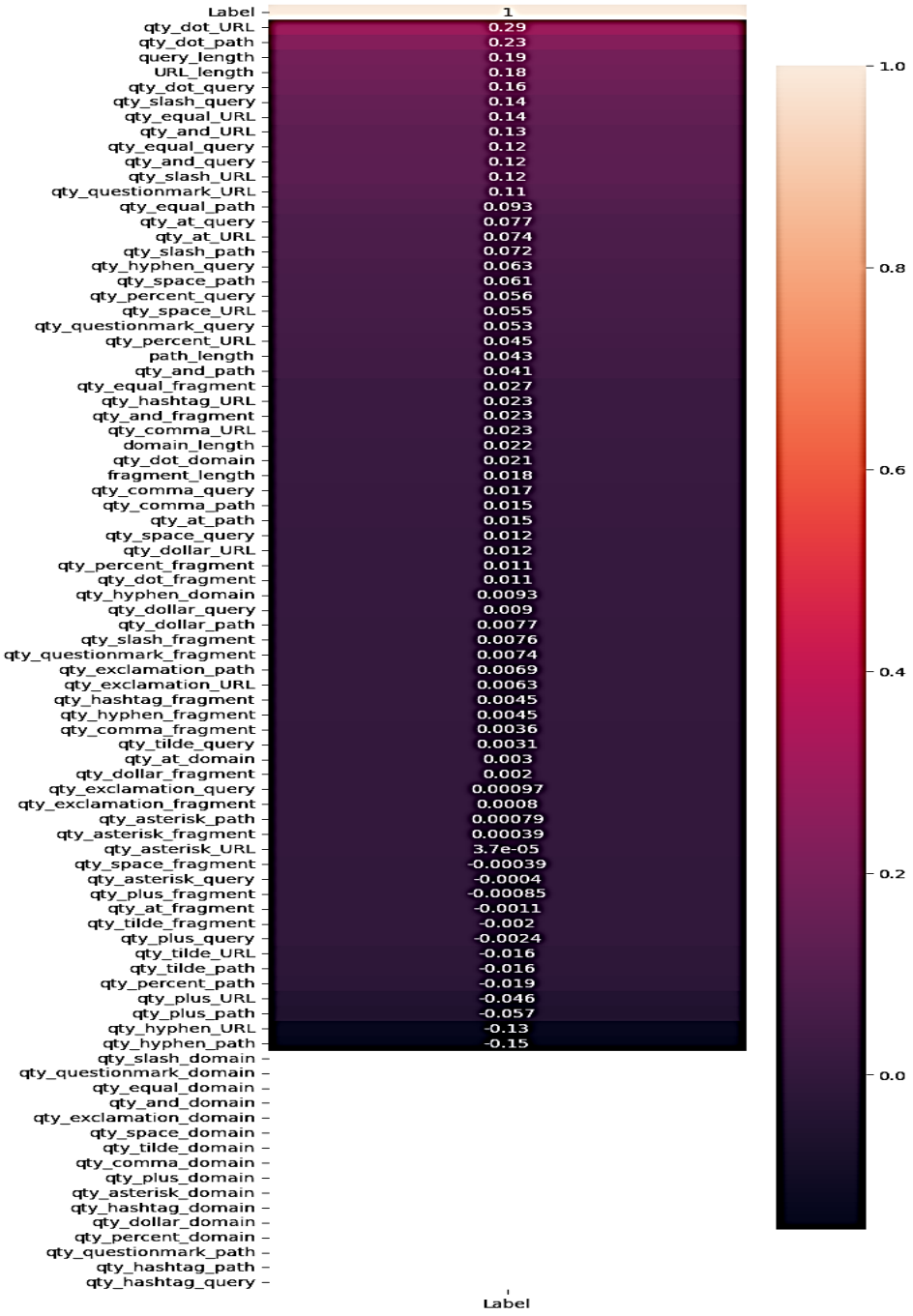

Figure 4.16. Heatmap result.

Heatmap-based feature selection is a powerful technique used to identify the most informative and relevant features in a dataset. By visualizing the correlation matrix between features, a heatmap provides a clear and intuitive representation of the relationships between variables. The color-coded heatmap allows for quick

dentification of strong correlations, both positive and negative, enabling efficient feature selection. Features that exhibit high correlation with the target variable or with other important features can be prioritized for further analysis, while irrelevant or redundant features can be disregarded. Heatmap-based feature selection aids introducing dimensionality, improving model interpretabilities. Figure 4.17 shows the result of heatmap in the data we have extracted.

We have applied feature selection in our work using one of the popular techniques. In the ever-evolving landscape of cybersecurity, the detection and prevention of phishing attacks are of paramount importance. Phishing URLs pose a significant threat to individuals and organizations alike, as they are often disguised as legitimate websites to deceive unsuspecting users. To combat this, feature selection techniques play a crucial role in identifying the most relevant indicators of phishing activity. One such technique is SelectKBest, which provides a powerful approach to selecting the most informative features for domain-based phishing URL analysis.

Understanding SelectKBest:

SelectKBest is a feature selection algorithm widely used in machine learning and data mining tasks. Its primary objective is to rank and select the most relevant features from a given dataset based on their statistical significance. This approach helps reduce the dimensionality of the data, removing irrelevant or redundant features and enhancing the performance of subsequent classification algorithms.

Applying SelectKBest to Phishing URL Domain Analysis:

When it comes to phishing URL analysis, the domain plays a crucial role in identifying potential threats. SelectKBest can be leveraged to select the most discriminative features from a dataset containing various domain-related attributes, such as domain length, the presence of hyphens or digits, and the presence of suspicious keywords. By quantifying the statistical relevance of these features, SelectKBest enables the

identification of the most influential factors in differentiating between legitimate and phishing URLs.

Feature Selection Process with SelectKBest:

Dataset Preparation: The first step involves gathering a labeled dataset comprising both legitimate and phishing URLs. Each URL should be represented by various domain-related attributes that capture the distinguishing characteristics.

Feature Extraction: Next, relevant features are extracted from the URLs. These features may include domain length, the number of subdomains, the presence of specific keywords or patterns, and other domain-centric attributes that could potentially differentiate between legitimate and phishing URLs.

Feature Scoring: In this step, SelectKBest applies a scoring function, such as chi-squared or mutual information, to assign a score to each feature. The score indicates the statistical significance of the feature in relation to the target variable (legitimate or phishing).

Feature Ranking: Once the features are scored, SelectKBest ranks them based on their scores. The top-k features with the highest scores are selected as the most informative and discriminatory features for subsequent analysis.

Model Training and Evaluation: The selected features are then used to train a classification model, such as a decision tree, random forest, or support vector machine (SVM). The model's performance is evaluated using appropriate evaluation metrics, such as accuracy, precision, recall, or F1-score, to assess its effectiveness in distinguishing between legitimate and phishing URLs.

Benefits and Limitations of SelectKBest:

SelectKBest offers several advantages in the context of phishing URL domain analysis. It reduces the dimensionality of the feature space, mitigates the curse of

dimensionality, and enhances the performance and interpretability of the subsequent classification models. Additionally, by selecting the most relevant features, SelectKBest can improve computational efficiency and reduce overfitting.

However, it is important to note that SelectKBest operates solely on the available features and does not consider potential interactions or dependencies among them. Therefore, it may not capture complex relationships between features, which could lead to the omission of important information. Additionally, SelectKBest relies on the assumption that the selected features are statistically significant, which may not always hold true in every dataset.

We set K equal to 10 and obtained important features that demonstrate a significant impact on the URL results. These features are highly correlated with the results. Therefore, the field with the highest score is as follows: url_length, qty_dot_url, qty_hyphen_url, qty_and_url, qty_dot_path, query_length, qty_dot_query, and qty_slash_query, as shown in Figure 4.18.

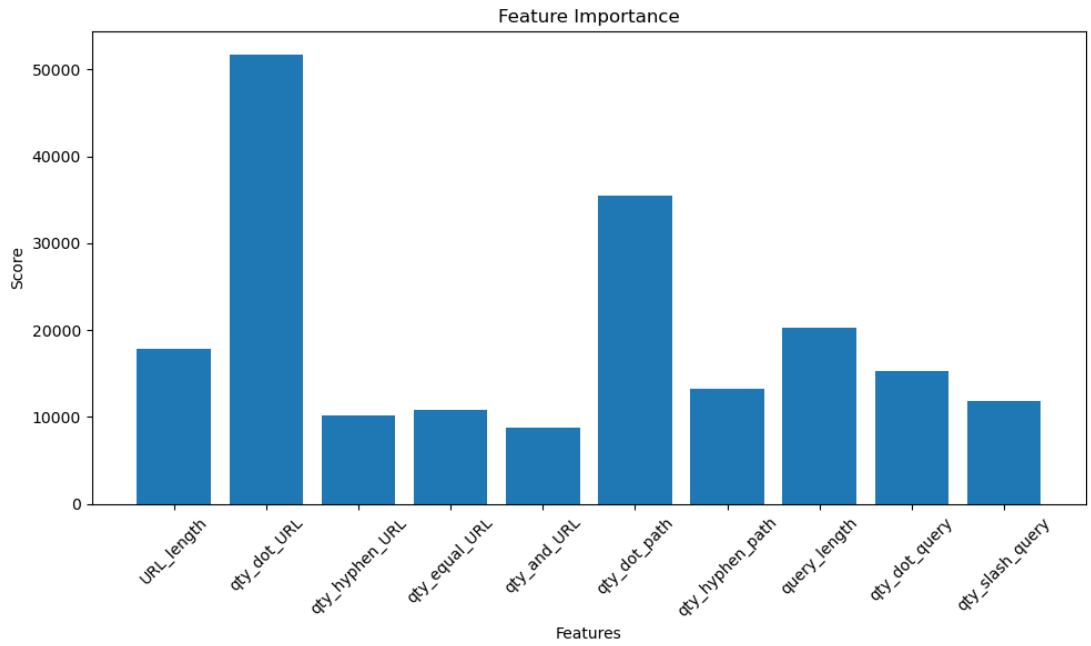

Figure 4.17. Results of the feature importance.

4.4. THE PROPOSED ALGORITHMS

In the next section, we will provide an in-depth explanation of the Core Classification Algorithm (CCA) as well as its improved form, the Improved Core Classification Algorithm (ICCA), which was introduced in the previous section [109].

4.4.1. Core Classification Algorithm (CCA)

Our proposed technique is derived from the KNN algorithm since it is based on comparing all of the points to a single point inside each class that is also known as the core. This comparison is done to determine how similar each point is to the core. When compared to all of the other points, this one singular point stands out. Because it possesses all of the characteristics that are associated with the class, or at least the vast majority of them, that particular point is considered to be an accurate reflection of the group as a whole. Therefore, the approach that was suggested is able to circumvent the challenges presented by the possibility of altering the results by adjusting the K parameter in the KNN algorithm. This is done in an attempt to find a suitable value that will offer a categorization that is as accurate as is feasible in the given circumstances.

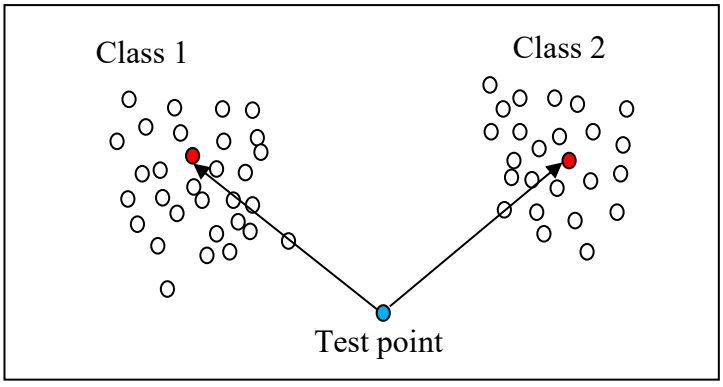

Figure 4.18. CCA algorithm classification [109].

In the case of liner categorization, which is shown in Figure 4.18, the test point has only been examined in conjunction with the Cores in both classes. Because of this, it will be placed in the category that is most closely associated with its fundamental

characteristics. The objective of the research was to hybridize the algorithm that was derived from KNN with one of the partition unsupervised learning algorithms (K-means). This is because the hybridization process gives the derived algorithm strength and enables it to work efficiently while dealing with a large number of domains and cases. Any algorithm used in machine learning has both strengths and flaws inside its underlying mechanism; as a result, it is only useful in some contexts while being of no use in others.

The concept of hybridizing two algorithms is required in many situations in order to improve performance or to increase efficiency, and what's more important is that it can be used effectively to overcome some of the flaws and challenges that are present in one of the two algorithms that were used to create the hybridization. Historically, hybrid algorithms have been developed according to the notion of maximizing the benefits of one algorithm while simultaneously improving the functionality and effectiveness of another algorithm. Yet, the results of our research indicate that it is of a greater significance when a fault in one algorithm is utilized to improve the performance of another algorithm. The core mechanism of the suggested algorithm (CCA) is based on the following three principles:

1. Modeling the KNN technique and locating a single Core that embodies all of the attributes unique to each class as an alternative to adjusting the classification result depending on the K-value generated by the KNN algorithm.
2. The clustering technique should be used to overcome the dataset distribution difficulties such as nonlinear classification, overlapping, or noise. All of these problems imitate the hidden layers in NNs; thus, the clustering approach should be used to solve them.
3. A number of rounds are necessary to create varying numbers of clusters reaching newer cores, similar to the strategy that NNs use, and this number varies depending on the outcomes of the K-means algorithm, which are notoriously unreliable.

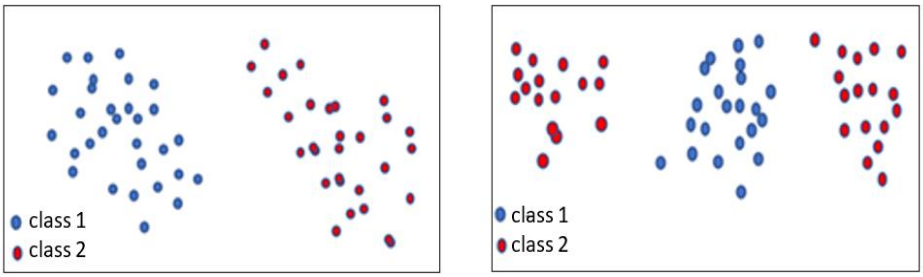

Figure 4.19. Linear and Non-linear classification [109].

Figure 4.19.a. the linear classification of the data distribution is shown on the left, where information is sorted into one of two categories according to how it was labeled. In circumstances when the core does not faithfully reflect all the points in the data class specified by the core, as seen in figure 4.19.b, it is necessary to combine the technique with clustering. It is hard to appropriately identify the data (particularly class 1) without introducing clustering which correctly aligns the distributions with the classes, as shown in the image, where classes are represented by C (C1 and C2) and clusters by K (K1 and K2).

4.4.1.1. Mathematical Formula

This part shows the basic math formula for the CCA algorithm without any hybridization of clustering to make it easier to understand and explain. The following formulas back up the mechanism of the proposed algorithm and show that it works. The first step of the algorithm is to make the distance matrix (D_M) for each class. Each row in D_M shows how far a point is from the other points in the same class. A new column is used to store the sum for each row. The row with the smallest sum shows the most similarities. This is also known as the Core point, which is a summary of most or all of a class's traits. Any test point will be put into a class based on how much it looks like the Cores of the classes. The scenario of three classes is shown symbolically as $c1, c2, c3$ from the original dataset (DS) and can be shown in sub-datasets as dsc1, dsc2, dsc3, with each class made up of a number of points (i, j, k), as follows:

$ds_{c1=}\{c_{1(p1)}, c_{1(p2)}, c_{1(p3)}, \dots, c_{1(pi)}\}$ ……..…… 4.1

$ds_{c2=}\{c_{2(p1)}, c_{2(p2)}, c_{2(p3)}, \dots, c_{2(pj)}\}$ …….. 4.2

$ds_{c3=}\{c_{3(p1)}, c_{3(p2)}, c_{3(p3)}, \dots, c_{3(pk)}\}$ ……..…. 4.3

The distance matrices for the three classes are each represented by the following notation: $(\mathrm{D_M})_{c1}$, $(\mathrm{D_M})_{c2}$, $(\mathrm{D_M})_{c3}$, with the following sizes: I * I j * j, and k * k. In the case of the $(\mathrm{D_M})_{c1}$, each row of the table comprises I cells, and these cells reflect the distances that separate every point (pn) in class 1 from the other points. In order to produce $(\mathrm{D_M})_{c2}$ and $(\mathrm{D_M})_{c3}$, the same method is used, as shown by the equations that follow:

$(\mathrm{D_M})_{c1} = \prod_{n=1}^{i} dis(c_{1(pn)}, ds_{c1})$ …….. 4.4

$(\mathrm{D_M})_{c2} = \prod_{n=1}^{j} dis(c_{2(pn)}, ds_{c2})$ …….. 4.5

$(\mathrm{D_M})_{c3} = \prod_{n=1}^{k} dis(c_{3(pn)}, ds_{c3})$ 4.6

The core vectors Core_v with sizes (i * 1), (j * 1), and (k * 1) are constructed by computing the sum of each row in DM reflecting the farthest extent that the point (row) may represent the features of its class. This results in the core vectors with sizes (i * 1), (j * 1), and (k * 1).

$(\mathrm{Cor\_}v)_{c1} = \prod_{n=1}^{i} \sum_{col=1}^{i} (\mathrm{D_M})_{c1}$ ……. 4.7

$(\mathrm{Cor\_}v)_{c2} = \prod_{n=1}^{j} \sum_{col=1}^{j} (\mathrm{D_M})_{c2}$ ……..... 4.8

$(\mathrm{Cor\_}v)_{c3} = \prod_{n=1}^{k} \sum_{col=1}^{k} (\mathrm{D_M})_{c3}$ ……...…. 4.9

The core of each class may be determined by picking the least value in Core_v, this core is distinguished by having the most in common with the other points that make

up the class. As a result, three cores denoted by the notations $(\mathrm{Co}r)_{c1}$, $(\mathrm{Co}r)_{c2}$, and $(\mathrm{Co}r)_{c3}$ are obtained for each class denoted by the notations c_1 c_2 c_3 respectively. As a result, test points can be assigned to the appropriate classes by achieving the highest level of similarity with their respective cores.

4.4.1.2. The Pseudo Code for CCA Algorithm

We will demonstrate this using a scenario in which a dataset is trained, and the results are divided into three categories. The similarity between a given point and the other points in its class is represented by the sum of its rows in the distance matrix, which is created as (DM)ci. For each class (ci), the best possible representation is found at the place with the greatest similarity-ty or the smallest sum. Instead of utilizing K closest neighbors like in the KNN method, the test points will be categorized based on their degree of similarity to these cores.

Algorithm: Algorithm for CCA

```
1: Input: in
2: Output: out
3: Initialization:
4: loop process
5: for i=1 to length of test_ds do
6:    find the dis between test_ds[i] and (Cor)cx
7:      if dis(1)  is minimum
8:            Classify test_ds[i] to class(1)
9:      else if  dis(2)  is minimum.
10:           Classify test_ds[i] to class(2)
11:     else
12:           Classify test_ds[i] to class(3)
13: end loop
14: return Class(x)
```

Think of S as a dataset with n points (S: =p1, p2..., pn) that are separated into three categories and represented as pi= (xi, ci) where xi is the point's feature vector and ci is the category to which it belongs.

Let's pretend x is a point that has to be placed in the right CCA class.

1. The first step is to determine the distance matrices between the three groups (DM)c1, DM)c2, and DM)c3.
2. Determine the center of the first, second, and third classes
3. Determine three values for x based on the calculated distance to all cores: Disadvantages 1, 2, and 3
4. Based on the minimal distance within each class, X will be placed in class (dis1, dis2, dis3)

The following pseudocode and table 1 notation illustrate these actions:

4.4.1.3. Hyperdization CCA Algorithm With K-means Algorithm

The utilization of real data in classification problems presents several challenges, including but not limited to nonlinear classification, overlap among classes, and the presence of outliers. To address these challenges, one may employ a clustering algorithm to enhance the flexibility of the CCA algorithm. This denotes a benefit of utilizing an algorithm within another algorithm to enhance its efficacy or address a specific problem. Employing clustering with CCA can be likened to incorporating hidden layers in Neural Networks (NN). Conversely, when partition clustering is applied to the same input dataset in a different NN simulation, it yields disparate outcomes. The K-means algorithm involves applying iteration to obtain distinct clusters at a predetermined number. The selection of initial centers is random and impacts the resulting clusters in each iteration. Consequently, the model shall undergo training in order to achieve greater levels of accuracy. Both methodologies are employed in addressing non-linear classification and other related predicaments.

According to the study's scenario, the training dataset separation structure comprises 80% of all datasets and is characterized by three classes that are each composed of two clusters. For example, Cor_c1 (k1) denotes the Core of cluster 1 within class 1, while Cor_c1 (k2) represents the Core of cluster 2 within class 1.

The general concept of the Core Classify Algorithm (CCA) is presented in Figure 4.20, which depicts a flowchart outlining three class scenarios divided into two clusters within each class. The testing dataset is represented by the variable "n," while "j" serves as a counter. The model is trained within each iteration process, with "I" serving as a counter for the number of iterations (ite). The accuracy and Cores are stored during this process. Ultimately, the cores of clusters exhibiting high accuracy shall be selected as suitable cores for an ideal model. The efficacy and scope of a classification algorithm are not contingent upon its precision in optimal scenarios, such as linear or discriminative classification. Rather, its effectiveness is determined by its ability to adjust to non-ideal classification scenarios. The proposed algorithm achieved higher classification accuracy by incorporating one of the most prevalent unsupervised learning algorithms. Furthermore, due to a characteristic of the K-means algorithm, varying outcomes that are not consistent are produced with each implementation. The simulation of training in neural networks involved the concept of updating the Cores within clusters, which distinguishes it from the weight updating process in neural networks. This difference lies in the varying strength of representation of clusters.

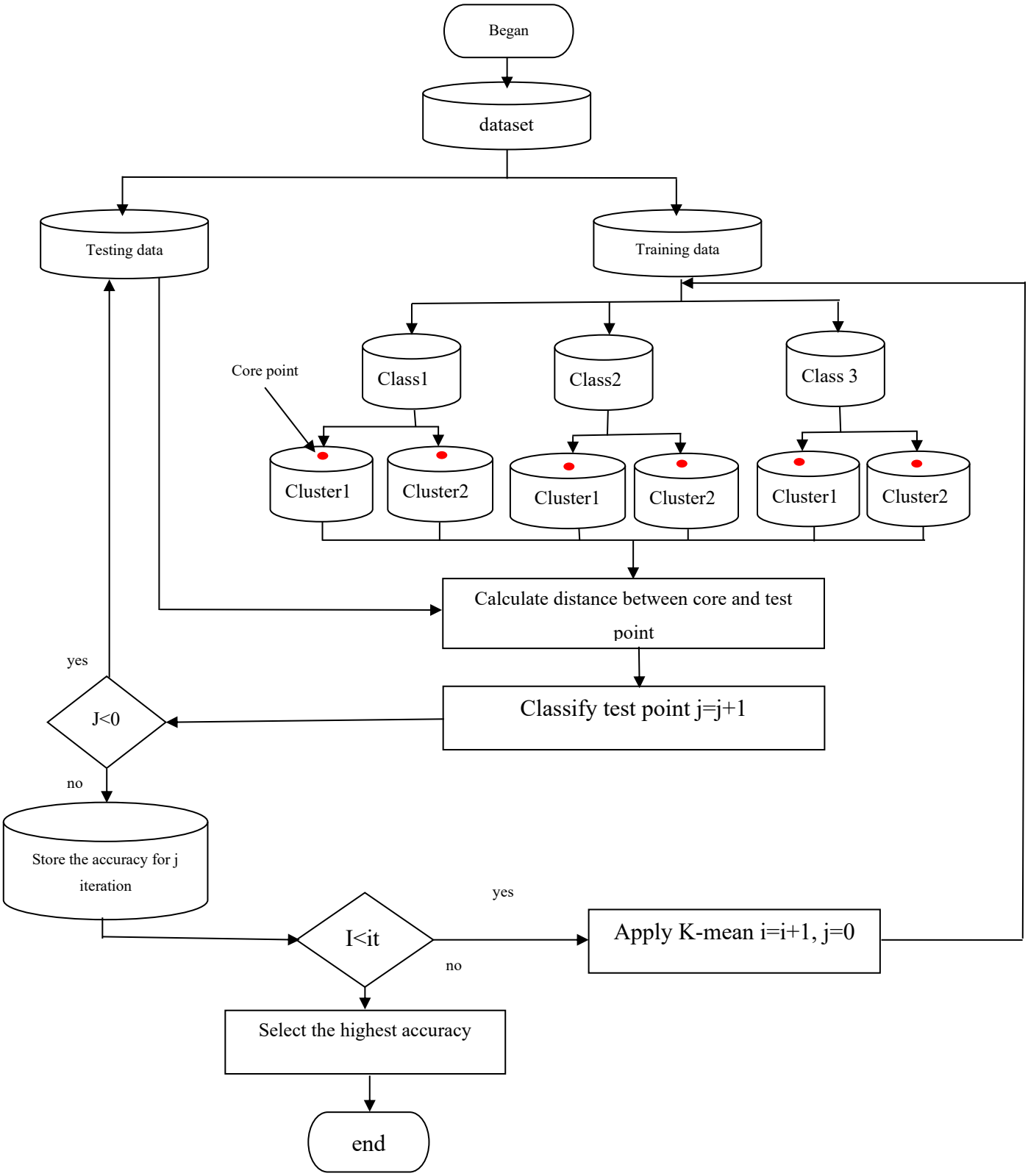

Figure 4.20. flowchart of CCA algorithm [109].

4.4.2. Improved Core Classification Algorithm (ICCA)

In this stage, we present a novel algorithm called the Improved Core Classification Algorithm (ICCA), which is an enhancement of the previous algorithm known as CCA. The goal of ICCA is to improve the accuracy and efficiency of core classification tasks.

The ICCA algorithm introduces a significant improvement by incorporating an active set (A_S) concept. The active set provides a more refined representation of the class, enabling better classification of data points. By leveraging similarity voting, ICCA measures the distance between data points using the well-known Euclidean distance metric. This distance-based approach allows ICCA to capture the similarity between points and make informed decisions about their classification.

One interesting aspect to consider is the variability in the output of K-means algorithms across different implementations. While this variability can sometimes pose a challenge, we turn it into an advantage. During the training model phase, we carefully utilize this property to enhance the overall accuracy of our algorithm. By leveraging the variations in the output of K-means algorithms, we fine-tune our model to adapt to different implementations, ensuring robustness and improved performance. To validate the effectiveness and applicability of ICCA, we will conduct experiments in two specific domains: phishing URLs and healthcare domains. Phishing URLs represent a significant challenge in modern cybersecurity, and accurately classifying them is crucial for protecting users from malicious activities. By testing ICCA on this domain, we aim to demonstrate its capability to effectively identify and classify phishing URLs, enhancing security measures.

Furthermore, we also plan to evaluate ICCA's performance in healthcare domains. Healthcare data often presents unique challenges due to its sensitive nature and complex characteristics. By applying ICCA to healthcare domains, we aim to demonstrate its ability to handle diverse data types, such as patient records, medical images, and diagnostic information. Achieving accurate and efficient classification in

healthcare settings can have far-reaching implications, including improved patient care, disease detection, and medical decision-making.

however, the introduction of the Improved Core Classification Algorithm (ICCA) builds upon the previous work in CCA and presents a novel approach to enhance core classification tasks. By incorporating the active set concept, leveraging similarity voting, and utilizing the variations in K-means algorithms, ICCA aims to achieve better accuracy and efficiency. Through rigorous testing and validation in the domains of phishing URLs and healthcare, we aim to demonstrate ICCA's effectiveness and its potential impact in various real-world applications Figure 4.21.

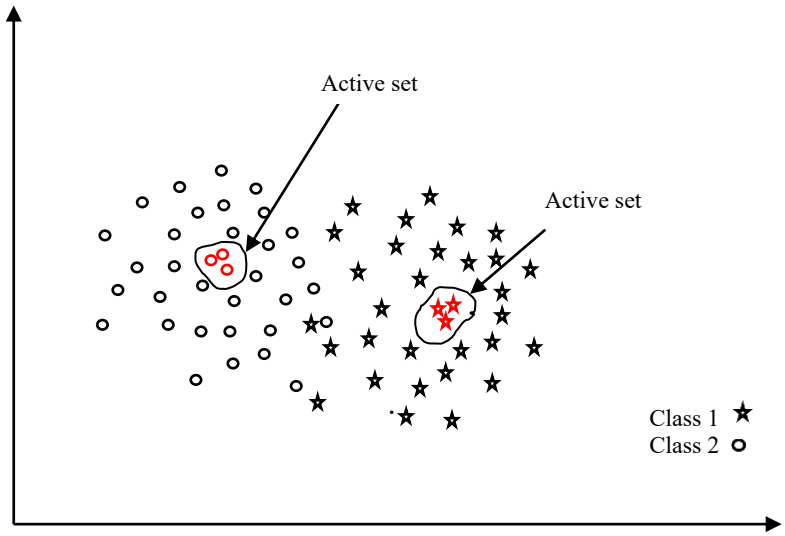

Figure 4.21. ICCA Classification Algorithm.

Hybridization, which involves combining two different algorithms into a single one, is essential for enhancing the functionality of both algorithms across various contexts. This approach proves to be useful in overcoming deficiencies and challenges that either of the algorithms may encounter. Historically, hybrid algorithms have been developed to leverage the strengths of one algorithm while improving the functionality and productivity of another. However, our study findings suggest that it is equally important, if not more so, to utilize the weaknesses of one algorithm to enhance the effectiveness of another algorithm.

The recommended algorithm, ICCA (Improved Core Classification Algorithm), is built upon the following key ideas that form its core mechanism:

1- The current work involves simulating the Canonical Correlation Analysis (CCA) technique to identify an active set (A_S) for each class. This active set is defined by the unique characteristics specific to that particular class. The technique aims to address the challenge of high data dispersion, providing a solution for this issue.
2- Clustering algorithms serve as powerful tools for efficiently solving problems associated with dataset distribution. These algorithms can effectively handle various challenges such as nonlinear classification, overlapping data points, and noise.

By combining these ideas, ICCA aims to create a hybrid algorithm that leverages the benefits of both the CCA technique and clustering algorithms. The integration of CCA allows ICCA to capture class-specific characteristics, while the clustering algorithms provide efficient solutions for dealing with complex dataset distributions.

Overall, hybridization plays a crucial role in improving the functionality and effectiveness of algorithms. By capitalizing on the strengths and weaknesses of different algorithms, ICCA represents an innovative approach to address various challenges in data analysis and classification tasks.

4.4.2.1. The Pseudo Code of The Proposed Algorithm

The approach for training a dataset and then later dividing the results into two unique groups will be shown. The building of the distance matrix, denoted by the symbol DMci, is carried out for each class. The summation of each row of the matrix reveals the degree of similarity that exists between a particular point (row) and the other points that belong to the same class. The distance matrix denotes the distance between the points. The active set A_S is a collection of entities that demonstrate the highest degree of interconnection. This collection may be found inside the current class and has been given its own name. This A_S is stored in the Active Set Matrix, and the Beta value is

established as a variable amount that indicates the number of points included inside the A_S. Both of these processes take place after the Active Set Matrix has been initialized. As a result, one may make the case that it offers the most realistic portrayal of the category that was indicated before. The data from the exam will be sorted into categories according to how closely they resemble the A_S.

Algorithm: Algorithm for ICCA

```
1: Input: training dataset
2: Output: classify the test point into its class
3: Initialization:
4: Find the  A_S for each class
5: loop process
6: for i=1 to length of test_ds do
7:    find the dis between test_ds[i] and  A_S objects
8:      if dis(1)  is minimum
9:           Classify test_ds[i] to class(1)
10:   else
11:          Classify test_ds[i] to class(2)
12: end loop
13: return Class(x)
```

1. Compute a matrix that represents the distance between the two classes.
2. Make the value of Beta the cardinality of the A_S, which is the number of objects included inside it, and set it to that value.
3. You may acquire the A_S matrix for each class by measuring the distance matrix. This is how you do it.
4. Determine the distance that separates each active object set from the variable x by computing the distance between the two.
5. In order to fulfill the requirements of the minimal distance criteria, the value of x has to be designated for the category that displays the shortest distance.

The following processes are shown in pseudocode format as they are provided below, and the accompanying symbols are illustrated as follows:

Clustering algorithms are a powerful tool that may be used to efficiently solve problems that are associated with the distribution of datasets. Some of these problems include, but are not limited to, nonlinear classification, overlapping, and noise.

Figure 4.22 is a flowchart that depicts the Improved Core Classify Algorithm (ICCA), which is designed to improve the classification process. The ICCA intends to improve the accuracy of classifications. This particular implementation of the method is intended to handle a situation in which there are two classes, each of which is separated into two clusters. "n" is the variable that stands in for the testing dataset, and "j" is the counter that is being used here. When training the model, the letter "I" serves as a counter for the total number of iterations. The accuracy and A_S are both tracked during each iteration of the process. The A_S values of clusters that have been determined to have a high level of accuracy are chosen to be the right values for the ideal model.

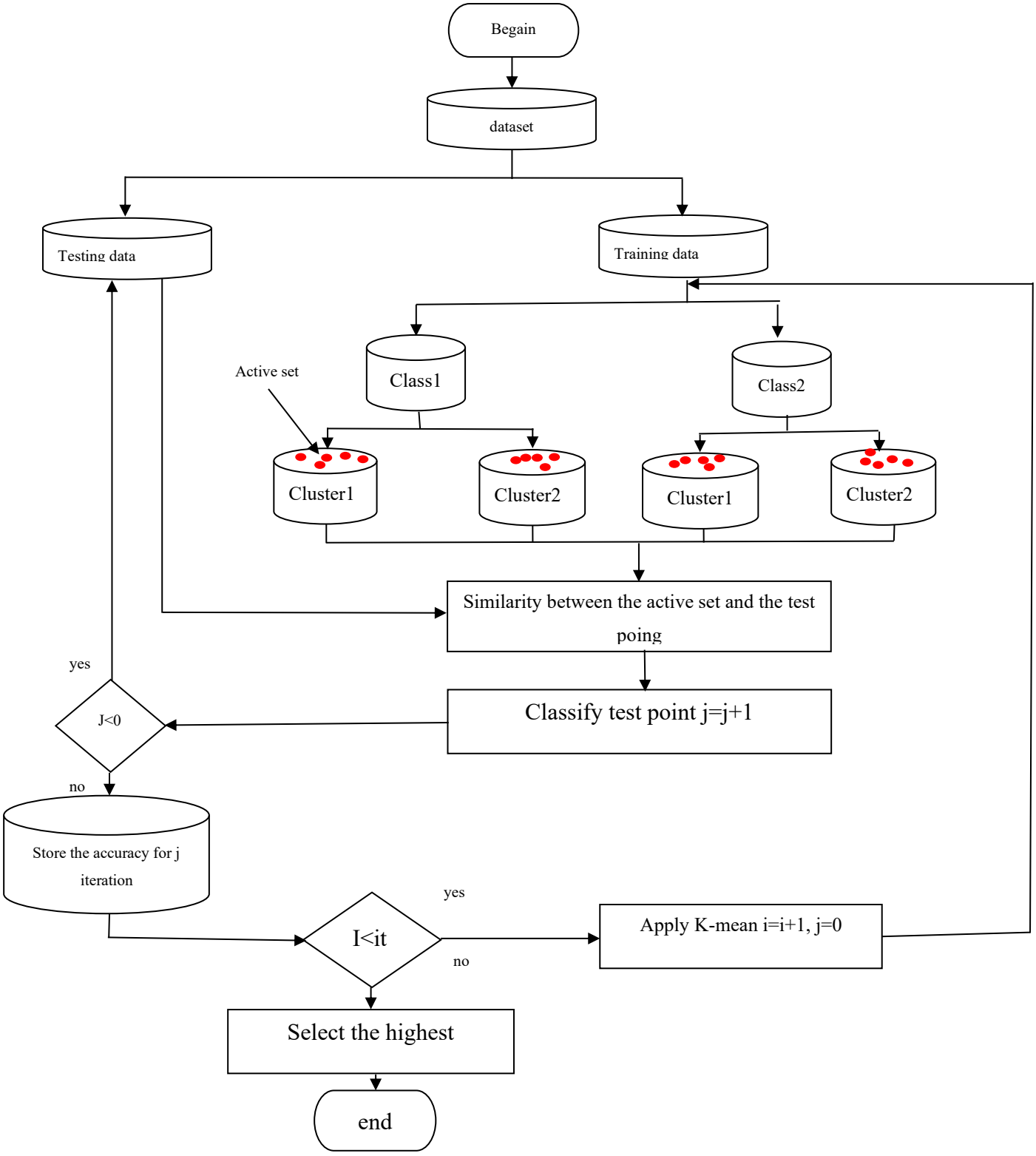

Figure 4.22. flowchart of ICCA algorithm.

PART 5

RESULTS AND DISCUSSIONS

5.1. EXPERIMENT AND RESULTS FOR CCA ALGORITHM:

The effectiveness and productivity of the CCA algorithm are evaluated by making use of five different real datasets coming from two distinct industries, namely Phisher URL and Healthcare, in addition to one linear classification synthetic dataset. These datasets are described in Table 5.1, and the evaluation of the CCA algorithm follows. The findings are going to be analyzed and spoken about, and then they're going to be compared to several different categorization systems. Many articles have presented various techniques for Web Phishing Attacks, such as Feature Selection, data preparation, and other topics associated with these attacks. As a result, phishing and healthcare datasets were acquired from their respective sources for this investigation [98-100].

Table 5.1. The data set description of the experiment [109].

Experiment. No.	Observations	Features	Domain
1	1000	3	synthetic
2	11055	30	Phisher URL
3	583	8	Healthcare
4	2000	11	Healthcare
5	3656	15	Healthcare

In Table, 5.2 The terms "Experiment 1-5" denote the five separate data sets that have been examined. CCA was performed on each dataset with a total of four iterations and four cluster indices (K-values). The findings of CCA show that the accuracy is at its highest for dataset 1, which represents a linear classification scenario, and that clustering is not required because the core of each class is adequately represented. This finding demonstrates that linear classification yields the highest level of accuracy. Studies 2 through 5, which could not be separated linearly, all shown greater accuracy when using the CCA, with some clusters doing much better than the others.

Table 5.2. Results of CCA with different numbers of (cluster, iteration) [109].

Experiment. No.	Iteration	K=1	K=2	K=3	K=6
1	50	100%	100%	100%	100%
	100	-----	100%	100%	100%
	300	-----	100%	100%	100%
	500	-----	100%	100%	100%
2	50	77.7%	**85.4%**	81.0%	82.7
	100	-----	80.7%	84.0%	83.6%
	200	-----	81.7%	84.0%	83.3%
	300	-----	80.4%	84.0%	83.7%
3	50	66.7%	59.2%	59.8%	58.6%
	100	-----	59.2%	61.3%	60.3%
	200	-----	59.2%	61.5%	62.7%
	300	-----	59.2%	**63.2%**	62.3%
4	50	59.9%	60.0%	60.7%	60.8
	100	-----	60.0%	61.0%	61.5
	200	-----	60.0%	61.0%	**61.8**
	300	-----	60.0%	61.0%	61.2
5	50	61.9%	67.3	76.9%	69.8%
	100	-----	68.7	76.9%	71.9%
	200	-----	75.1	76.9%	71.1%
	300	-----	75.1	**77.2%**	75.9%

In table 5.3. We determined the efficacy of the CCA by doing the following analyses: calculating the F1-score; determining the level of accuracy and recall; and analyzing the data using the confusion matrix. We examined four datasets (numbered 1–4), and discovered that CCA has yielded inconsistent results depending on the kind of data. All of the experiments were run with a maximum of two iterations each.

Table 5.3. Results of F1-Score, Precision, and Recall for CCA [109].

Dataset	Performance Matrix	CCA with 0 Cluster	CCA with 2 Cluster	CCA with 3 Cluster	CCA with 0 Cluster
1	precision	76.24	75.73	80.22	76.35
1	Recall	77.01	76.94	80.17	77.20
1	F1 Score	76.62	75.33	80.17	76.77
2	Precision	95.22	50.54	53.85	62.87
2	Recall	62.05	50.70	55.00	56.63
2	F1 Score	60.60	50.62	54.42	64.22
3	Precision	55.59	58.18	62.00	59.07
3	Recall	55.68	58.24	62.08	59.08
3	F1 Score	55.64	58.21	62.04	59.07
4	Precision	54.19	59.94	59.76	52.89
4	Recall	52.28	56.34	55.93	51.52
4	F1 Score	53.20	58.08	57.78	52.20

5.1.1. Comparison CCA Algorithm and Other Well Know Algorithms

The purpose of this part is to show the characteristics of the proposed algorithm as well as its areas of strength and weakness by comparing the performance of the proposed algorithm to that of other comparable and well-known machine learning methods. The accuracy of the model generated by the suggested method is evaluated and contrasted with that of Random Forest, SVM, and Decision Trees. Table 5.4 displays the results of the experiments that were conducted using the three different classification methods, including CCA. As compared to experiment 1, in which all algorithms functioned perfectly when subjected to linear classification, it would seem that the accuracy of the model is worse in all non-linear classification tests, such as those seen in experiments 2-5. In terms of accuracy, the RF method was the least precise, followed by the SVM method, then the DT method, and lastly the CCA method. The results also demonstrate that the recommended algorithm suffers from the same problems with data distribution as the other three methodologies that were investigated. Experiment 1 exhibits the highest level of accuracy that can be achieved with the model, whilst experiment 4 demonstrates the lowest level.

Table 5.4. Results of compare CCA and other algorithms [109].

NO	RF	SVM	DT	CCA	High Accuracy
1	100%	100%	100%	100%	All
2	97.3%	93.1%	96.5%	84.0%	RF
3	73.6%	71.3%	69.5%	63.2%	RF
4	71.3%	64.3%	64.7%	61.8%	RF
5	84.2%	84.7%	75.5%	77.2%	SVM
Avg	85.4%	82.7%	81.2%	77.2%	

5.2. EXPERIMENT AND RESULTS OF ICCA ALGORITHM

In this study, we put the ICCA technique to the test by applying it to four datasets that originate from two distinct application domains: phishing URL and healthcare. These datasets come from the Phisher URL and Healthcare application domains, respectively. The purpose of this research is to evaluate how effective and practical the approach is. According to what is shown in Table 5.5, the results are going to be

studied and discussed, and after that, they are going to be compared to a number of other classification schemes. Several publications have been published that offer various techniques for Web Phishing assaults. These techniques include Feature Selection, the preparation of data, and other issues that are linked with these assaults. As a direct consequence of this, datasets related to phishing and healthcare were obtained from their respective sources in order to conduct this analysis[96, 98-100].

Table 5.5. Data sets description of experiments.

Experiment. No.	**Observations**	**Features**	**Domain**
1	11055	30	Phisher URL
2	583_2	8	Healthcare
3	2000_4	11	Healthcare
4	3656_9	15	Healthcare

The following tables provide the findings obtained from doing an analysis on four distinct datasets; these datasets are denoted by the numbers "1-4." In any case, the following experiments have shown that the results of the ICCA have experienced a substantial improvement. This is especially obvious when the number of cores employed in each class is doubled.

In this study, the ICCA was examined by using the confusion matrix; moreover, the F1-score, precision, and recall were all calculated. We examined datasets 1 through 4, and ICCA supplied us with a variety of results, depending on whatever dataset we were analyzing. In each experiment, we carried out 20 iterations, 50 iterations, and 100 iterations accordingly.

The ICCA was evaluated without making use of clustering, as shown in Table 5.6, and the Beta was found to be equal to 5.

Table 5.6. Results of ICCA where β = 5, Number of Cluster =0.

Data set	Accuracy	Precision	recall	F1 score
1	77.12	77.31	76.08	76.08
2	68.96	67.44	66.83	65.56
3	60.83	60.91	60.90	60.10
4	62.81	54.52	52.42	61.21

By using the Cluster, setting the Beta to Table 5.7, and iterating until we reach a total of 20, 50, and 100, respectively. When the number of iterations was increased, the data indicated an increase, and the results were much better when clustering was utilized as opposed to when it was not employed.

Table 5.7. Results of ICCA where β = 5, Number of Cluster = 2, iteration = 20,50,100.

Data set	iteration	Accuracy	Precision	recall	F1 score
1	20	78.47	70.42	71.77	71.09
	50	80.95	71.37	72.68	72.02
	100	79.53	67.88	71.15	69.47
2	20	61.49	56.94	55.60	56.26
	50	65.52	62.98	61.25	62.10
	100	67.24	62.66	60.65	61.64
3	20	62.33	51.75	54.35	53.02
	50	60.17	56.21	56.21	56.21
	100	61.67	60.60	61.50	61.05
4	20	80.49	60.91	56.55	58.65
	50	81.22	61.99	57.10	59.45
	100	80.22	60.95	56.95	58.88

In addition, in table 5.8 the findings have been much better once the cluster size was raised to three. This has resulted in a more evenly distributed dataset, which in turn has led to superior outcomes.

Table 5.8. Results of ICCA where β = 5, Number of Cluster = 3, iteration = 20,50,100.

Data set	iteration	Accuracy	Precision	recall	F1 score
1	20	83.36	80.45	80.03	80.24
	50	85.35	74.39	75.44	74.91
	100	88.15	75.84	76.72	76.28
2	20	72.41	55.06	67.13	60.50
	50	75.86	64.20	65.38	64.79
	100	68.97	61.24	60.33	60.78
3	20	67.00	44.07	44.50	44.64
	50	61.50	57.07	57.10	57.09
	100	64.50	62.91	63.50	63.21
4	20	72.23	51.87	57.22	51.54
	50	80.71	47.35	48.71	48.02
	100	80.75	57.33	54.16	55.70

In Table 5.9. The Beta value is now 9 and the clustering was not provided, and the results for this table indicate improved results that are more in line with the method that was provided in comparison to when the lambda value was equal to 5.

Table 5.9. Results of ICCA where β = 9, Number of Cluster = 0.

Data set	Accuracy	Precision	recall	F1 score
1	78.66	77.36	77.29	76.31
2	70.69	68.62	66.67	67.63
3	61.83	61.87	61.98	61.92
4	63.45	58.80	55.01	56.84

In table 5.10. We may conclude that using the cluster and raising the value of Beta yields more accurate results.

Table 5.10. Results of ICCA where β = 9, Number of Cluster = 2, iteration = 20,50,100.

Data set	iteration	Accuracy	Precision	recall	F1 score
1	20	80.52	71.62	72.84	72.22
	50	88.79	71.38	72.56	71.97
	100	89.32	70.43	71.59	71.00
2	20	63.79	64.38	62.43	63.39
	50	68.39	67.18	65.25	66.20
	100	71.26	62.51	59.88	61.17
3	20	60.83	60.81	60.80	60.81
	50	63.33	58.76	58.87	58.81
	100	63.33	55.11	55.29	55.20
4	20	79.58	60.11	56.05	58.01
	50	68.28	61.66	57.58	59.55
	100	79.76	64.19	58.82	61.39

Finally, in table 5.11, the greatest values of Beta and clustering have been used, as well as the best outcomes in practically all datasets.

Table 5.11. Results of ICCA where β = 9, Number of Cluster = 3, iteration = 20,50,100.

Data set	iteration	Accuracy	Precision	recall	F1 score
1	**20**	**83.93**	**76.87**	**81.00**	**78.89**
	50	**80.42**	**80.66**	**80.13**	**80.40**
	100	**90.32**	**86.41**	**89.17**	**87.77**
2	20	70.69	59.04	66.34	62.47
	50	71.55	62.29	63.93	63.10
	100	73.28	57.64	61.71	59.60
3	**20**	**67.75**	**62.60**	**62.64**	**62.62**
	50	**66.25**	**56.79**	**58.33**	**57.55**
	100	**68.00**	**66.54**	**66.62**	**66.58**
4	20	81.81	57.04	53.72	55.33
	50	81.53	57.11	53.48	55.23
	100	81.67	61.51	57.11	59.23

5.2.1. Comparison With Other Classification Algorithms

In this section, we conduct a performance comparison of our proposed algorithm with other well-known and comparable machine learning algorithms. The objective is to highlight the distinctive features of our algorithm and identify any inherent limitations. We evaluate the model accuracy of our proposed approach in comparison to the Core Classification algorithm, Support Vector Machines (SVM), and Decision Trees.

The experimental results, including those obtained from ICCA and the other classification methods, are presented in Table 5.12. These results serve as evidence of the power and performance of our algorithm, demonstrating its superiority over both its predecessor, the Core Classification algorithm, as well as the SVM and DT algorithms as it shown in figure 5.1.

By comparing the model accuracy achieved by each algorithm, we can clearly observe the advantages of our proposed approach. It outperforms the Core Classification algorithm, SVM, and Decision Trees, indicating its ability to effectively handle the given classification task.

These findings emphasize the significance of our proposed algorithm and its potential applications in various domains. Furthermore, they shed light on its ability to overcome the limitations of existing algorithms and provide superior performance in terms of accuracy.

Table 5.12. the comparison between ICCA and other classification algorithms.

NO	CCA	SVM	DT	ICCA	High accuracy
1	84.0%	93.1%	96.5%	90.32%	DT
2	63.2%	71.3%	69.5%	75.86%	ICCA
3	61.8%	64.3%	64.7%	68.0%	ICCA
4	77.2%	84.7%	75.5%	81.81%	SVM
Avg	77.2%	82.7%	81.2%	78.99%	

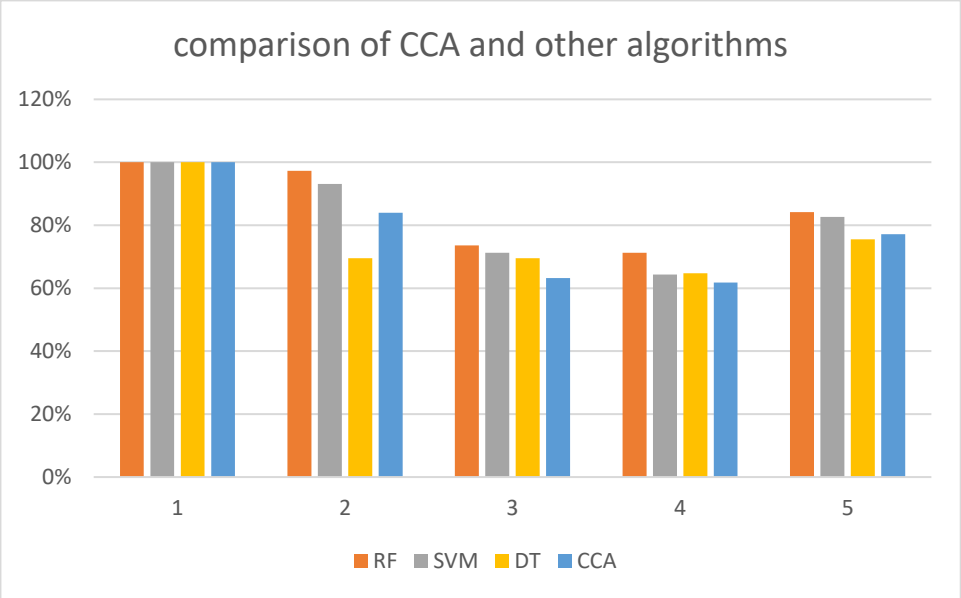

Figure 5.1. Comparison of CCA and other algorithms.

In Figure 5.2, the experimental results of applying the ICCA and CCA algorithms using four different datasets are presented. The purpose of this experiment is to assess the performance of these algorithms and determine their effectiveness. Upon analyzing the results, it becomes evident that the ICCA algorithm outperforms its predecessor significantly.

The experiment involved the utilization of four diverse datasets, providing a comprehensive evaluation of the algorithms' capabilities. By examining the outcomes,

we can gain valuable insights into the comparative strengths and weaknesses of the ICCA and CCA algorithms.

Moreover, it is worth noting that the ICCA algorithm demonstrates superior performance when compared to its predecessor. The significant advancements offered by the ICCA algorithm make it a promising choice for various applications. This observation suggests that further research and development in this area can lead to substantial improvements in the field.

By extending the analysis and considering additional factors such as computational efficiency, scalability, and robustness, we can gain a more comprehensive understanding of the ICCA algorithm's capabilities. This information will be crucial for researchers and practitioners seeking to leverage its potential benefits.

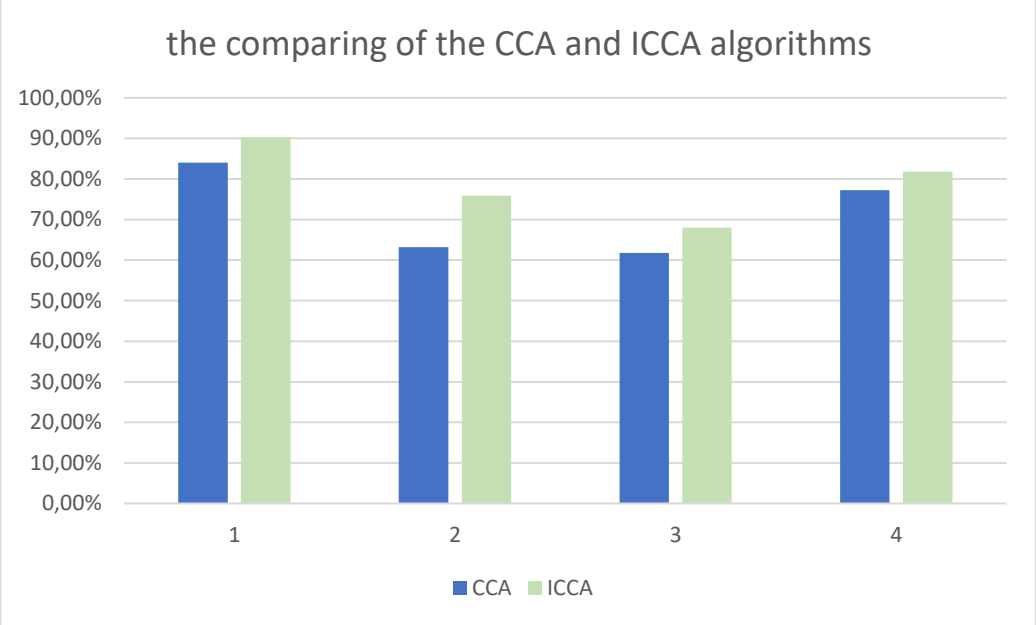

Figure 5.2. Comparing the Accuracy Performance of CCA and ICCA Algorithms.

The proposed model was applied in this study to detect phishing URLs. The approach involved feature engineering to extract relevant features from the URL and feature selection to identify the most important features that are highly correlated with the results. The SelectKBest method was employed, with a value of K set to 10. Subsequently, various machine learning algorithms, including our proposed algorithm, were applied, and the results obtained were compared to those of well-known

algorithms. Our proposed algorithm demonstrated favorable performance in comparison.

In Table 5.13, we have utilized a dataset comprising our extracted features. This dataset consists of 10 distinct features, each representing a specific characteristic of the URLs under consideration. To ensure a comprehensive analysis, we collected a substantial sample size, consisting of 549346 instances. This large-scale dataset allowed us to obtain statistically significant results and draw robust conclusions regarding the performance of our proposed model.

Table 5.13. The final results of the model.

Algorithm	Accuracy
CCA	70.12
SVM	75.5
DT	74.11
ICCA	75.7

However, as shown in Figure 5.3, our model has achieved superior results compared to the other examined algorithms.

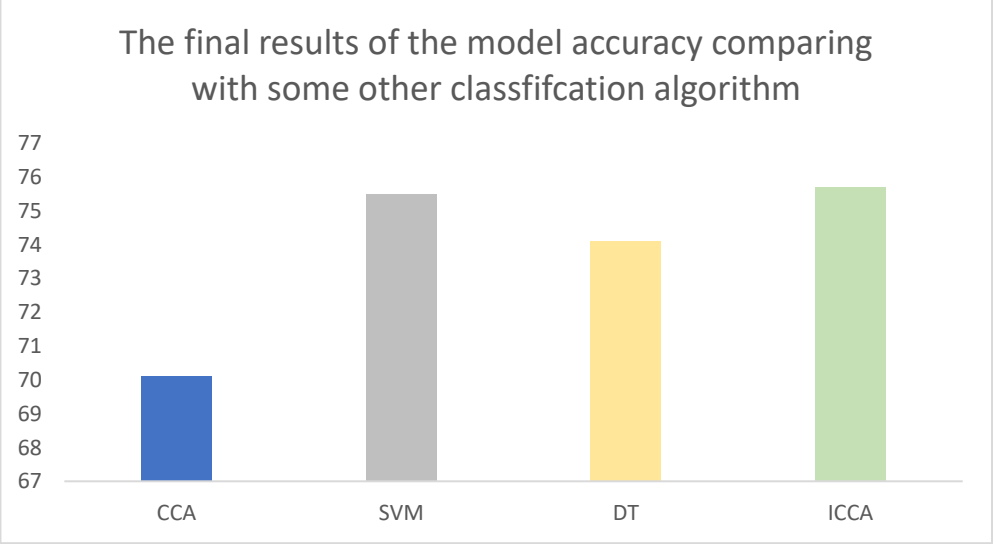

Figure 5.3. The final results of the model accuracy comparing with some other classfifcation algorithm..

PART 6

CONCLUSION

Our work comprises three main parts. In the first part of this study, we introduced a novel classification algorithm that incorporates clustering to address data distribution challenges in classification. Our proposed CCA algorithm achieved higher accuracy by leveraging the unique characteristics of the K-means algorithm, which dynamically adjusts the data distribution with each iteration. This approach holds the potential to pave the way for improved and more efficient computational methods in dealing with datasets containing mixed distributions.

This advancement serves as a stepping stone towards enhancing the accuracy and effectiveness of computational approaches for datasets with mixed distributions. To evaluate our proposed algorithm, we conducted four tests on two datasets from different domains: Phishing URL and Healthcare. Additionally, we performed one experiment on a synthetic dataset.

The findings from our experiments demonstrate that the incorporation of clustering techniques enhances the accuracy of the classification model. This improvement can be attributed to the fact that each data point is associated with an appropriate number of clusters, resulting in better overall performance. Furthermore, increasing the number of iterations for the K-means algorithm may further enhance the accuracy.

While the initial experiment using linear classification yielded excellent results for all methods, the accuracy of the algorithms varied when applied to the remaining four trials involving datasets with mixed distributions. This variability was mitigated by the elimination of the k parameter's changing values. Additionally, addressing the issue of missing values in the dataset is crucial. Various methods, such as data mining techniques and statistical measures like One-Way ANOVA, can be employed to handle

this problem. Using alternative similarity measures, such as One-Way ANOVA, instead of relying solely on Euclidean distance to express similarity between test points and the core of each cluster, can lead to improved results.

In the second part of our study, we proposed an enhanced Core Classification Algorithm (ICCA) that utilizes the most informative features to improve data representation. By leveraging the active set approach and refining the K-means clustering algorithm, we achieved substantial improvements in classification accuracy. In comparison, our previous algorithm, CCA, demonstrated only moderate success.

Our experimental results clearly indicate that ICCA outperforms CCA, particularly when working with high-dimensional datasets. In certain cases, ICCA even outperforms well-established algorithms like Random Forest (RF) and Decision Trees (DT). However, it is important to acknowledge that ICCA still has some limitations, with its time-consuming nature being the most prominent one. This can pose challenges when dealing with large-scale datasets. Addressing this limitation should be a priority for future work in this area.

In general, the ICCA method we proposed holds great potential for enhancing classification accuracy. It can be a valuable tool in various real-world applications, including fraud detection, medical diagnosis, and spam filtering.

Finally, we focused on studying the features of URLs to extract important characteristics for phishing detection. However, as phishers have become more sophisticated, we employed feature engineering techniques to extract relevant features and implemented the model on a real dataset consisting of 507,195 instances. Through the application of machine learning algorithms, our proposed approach demonstrated strong and accurate performance

REFERENCES

1. Internet: ”Apwg. phishing activity trends report”. 2022 [cited 2023 15 February 2023]; Available from: **https://docs.apwg.org/reports/apwg_trends_report_q3_2022.pdf?_ga=2.90908314.1625966364.1676415213-2057748636.1669818538&_gl=1*zntqps*_ga*MjA1Nzc0ODYzNi4xNjY5ODE4NTM4*_ga_55RF0RHXSR*MTY3NjQ1NTI2My44LjEuMTY3NjQ1NTI3Mi4wLjAuMA.**

2. Marsland, S., ”Machine learning: an algorithmic perspective”. 9-10 (2015) : ***CRC press.***

3. Burkov, A., ”*The hundred-page machine learning book*”. Vol. 1. 2019: ***Andriy Burkov Quebec City***, QC, Canada.

4. Ali, W., ”Phishing website detection based on supervised machine learning with wrapper features selection”. ***International Journal of Advanced Computer Science and Applications***, 2017. **8**(9).

5. Wei, B., et al., ”*A deep-learning-driven light-weight phishing detection sensor*”. ***Sensors***, 2019. **19**(19): p. 4258.

6. Mohammad, R.M., F. Thabtah, and L. McCluskey, ” Predicting phishing websites based on self-structuring neural network”. ***Neural Computing and Applications***, 2014. **25**: p. 443-458.

7. Folino, G., A. Forestiero, and G. Spezzano, ”A Jxta Based Asynchronous Peer-to-Peer Implementation of Genetic Programming”. ***J. Softw***., 2006. **1**(2): p. 12-23.

8. Cicirelli, F., et al., ”Transparent and efficient parallelization of swarm algorithms”. ***ACM Transactions on Autonomous and Adaptive Systems (TAAS)***, 2016. **11**(2): p. 1-26.

9. Jabbar, M., S. Samreen, and R. Aluvalu, ”The future of health care: Machine learning”. ***International Journal of Engineering & Technology***, 2018. **7**(4.6): p. 23-25.

10. Mohamed, G., et al., ”An Effective and Secure Mechanism for Phishing Attacks Using a Machine Learning Approach”. ***Processes***, 2022. **10**(7): p. 1356.

11. Alhogail, A. and A. Alsabih, ”Applying machine learning and natural language processing to detect phishing email”. ***Computers & Security***, 2021. **110**: p. 102414.

12. Alshingiti, Z., et al., ”A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN”. ***Electronics***, 2023. **12**(1): p. 232.

13. Zamir, A., et al., ”Phishing web site detection using diverse machine learning algorithms”. ***The Electronic Library***, 2020. **38**(1): p. 65-80.

14. Aalla, H.V.S., N.R. Dumpala, and M. Eliazer, ”Malicious URL Prediction Using Machine Learning Techniques”. ***Annals of the Romanian Society for Cell Biology***, 2021: p. 2170-2176.

15. Ozcan, A., et al., ”A hybrid DNN–LSTM model for detecting phishing URLs”. ***Neural Computing and Applications***, 2021: p. 1-17.

16. Martín-Valdivia, M.-T., et al., ”Sentiment polarity detection in Spanish reviews combining supervised and unsupervised approaches”. ***Expert Systems with Applications***, 2013. **40**(10): p. 3934-3942.

17. Rakić, M., et al., ”icobrain ms 5.1: Combining unsupervised and supervised approaches for improving the detection of multiple sclerosis lesions”. ***NeuroImage: Clinical***, 2021. **31**: p. 102707.

18. Mazumder, A.M.R., et al. ”Network intrusion detection using hybrid machine learning model”. in *2021* ***International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)***. 2021. IEEE.

19. Omta, W.A., et al., ”Combining Supervised and Unsupervised Machine Learning Methods for Phenotypic Functional Genomics Screening”. ***SLAS Discovery***, 2020. **25**(6): p. 655-664.

20. Martinez-de-Pison, F.J., et al. ”Hybrid methodology based on bayesian optimization and ga-parsimony for searching parsimony models by combining hyperparameter optimization and feature selection”. in ***Hybrid Artificial Intelligent Systems: 12th International Conference, HAIS*** *2017, La Rioja, Spain, June 21-23, 2017, Proceedings 12*. 2017. Springer.

21. Ippolito, M., J. Ferguson, and F. Jenson, ”Improving facies prediction by combining supervised and unsupervised learning methods”. ***Journal of Petroleum Science and Engineering***, 2021. **200**: p. 108300.

22. Chuang, P.J. and S.H. Li. ”Network Intrusion Detection using Hybrid Machine Learning”. in *2019* ***International Conference on Fuzzy Theory and Its Applications (iFUZZY)***. 2019.

23. Chkirbene, Z., et al. ”Hybrid machine learning for network anomaly intrusion detection”. in *2020* ***IEEE international conference on informatics, IoT, and enabling technologies (ICIoT)***. 2020. IEEE.

24. Ma, E.-Y., et al., ”Combined unsupervised-supervised machine learning for phenotyping complex diseases with its application to obstructive sleep apnea”. ***Scientific Reports***, 2021. **11**(1): p. 4457.

25. Gan, H., et al., ”Using clustering analysis to improve semi-supervised classification”. ***Neurocomputing***, 2013. **101**: p. 290-298.

26. Sapkota, N., et al. ”Data summarization using clustering and classification: Spectral clustering combined with k-means using nfph”. in *2019* ***International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)***. 2019. IEEE.

27. Agrawal, U., et al., ”Combining clustering and classification ensembles: A novel pipeline to identify breast cancer profiles”. ***Artificial intelligence in medicine***, 2019. **97**: p. 27-37.

28. Peng, G., S. Tang, and Y. Zhang. ”A combined clustering algorithm for the classification of electrical equipment's family defects”. in *2016 China* ***International Conference on Electricity Distribution (CICED)***. 2016. IEEE.

29. Soheili, M.R., et al. ”Merging clustering and classification results for whole book recognition”. in *2017 10th* ***Iranian Conference on Machine Vision and Image Processing (MVIP)***. 2017. IEEE.

30. Orhan, U., M. Hekim, and M. Ozer, ”EEG signals classification using the K-means clustering and a multilayer perceptron neural network model”. ***Expert Systems with Applications***, 2011. **38**(10): p. 13475-13481.

31. He, X., et al. ”Combining clustering coefficient-based active learning and semi-supervised learning on networked data”. in *2010* ***IEEE International Conference on Intelligent Systems and Knowledge Engineering***. 2010. IEEE.

32. Chakraborty, T. *Ec3*: ”Combining clustering and classification for ensemble learning”. in *2017* ***IEEE international conference on data mining (ICDM)***. 2017. IEEE.

33. Aissaoui, O.E.L., et al., ”Combining supervised and unsupervised machine learning algorithms to predict the learners’ learning styles”. ***Procedia Computer Science***, 2019. **148**: p. 87-96.

34. Song, I.J. and W. Heo, ”Improving insurers’ loss reserve error prediction: Adopting combined unsupervised-supervised machine learning techniques in risk management”. ***The Journal of Finance and Data Science***, 2022. **8**: p. 233-254.

35. Mitchell, T.M. and T.M. Mitchell, ”Machine learning”. 8-16 (Vol. 1. 1997): ***McGraw-hill New York***.

36. Sag, M., ”The new legal landscape for text mining and machine learning”. ***J. Copyright Soc'y USA***, 2018. **66**: p. 291.

37. Hu, J., et al., ”Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning”. ***IEEE Transactions on Vehicular Technology***, 2020. **69**(12): p. 14413-14423.

38. Yoosefzadeh-Najafabadi, M., et al., ”Application of machine learning algorithms in plant breeding: predicting yield from hyperspectral reflectance in soybean”. ***Frontiers in plant science***, 2021. **11**: p. 624273.

39. Rajasekaran, K. and P. Saravanan, ”Conceptual methodology on machine learning and types of learning algorithms”. ***JAC: A JOURNAL OF COMPOSITION THEORY***, 2020. **13**: p. 233-249.

40. Russell, S.J., ”Artificial intelligence a modern approach”. 2010: ***Pearson Education, Inc***.

41. Vapnik, V.N. and V.N. Vapnik, ”Introduction: Four periods in the research of the learning problem”. ***The nature of statistical learning theory***, 2000: p. 1-15.

42. Maity, A., ”Supervised classification of radarsat-2 polarimetric data for different land features”. ***arXiv preprint arXiv***:1608.00501, 2016.

43. Hinton, G. and T.J. Sejnowski, ”Unsupervised learning: foundations of neural computation”. 1999: ***MIT press***.

44. Sinaga, K.P. and M.-S. Yang, ”Unsupervised K-means clustering algorithm”. ***IEEE access***, 2020. **8**: p. 80716-80727.

45. Berry, M.W., A. Mohamed, and B.W. Yap, ”Supervised and unsupervised learning for data science”. 2019: ***Springer***.

46. Celebi, M.E. and K. Aydin, ”Unsupervised learning algorithms”. 16-25 (Vol. 9. 2016): ***Springer***.

47. Anandkumar, A., et al., ”Tensor decompositions for learning latent variable models”. ***Journal of machine learning research***, 2014. **15**: p. 2773-2832.

48. Zhou, Z.-H., ”A brief introduction to weakly supervised learning”. ***National science review***, 2018. **5**(1): p. 44-53.

49. Langley, P., ”Elements of machine learning”. 5-19(1996): ***Morgan Kaufmann***.

50. Sen, P.C., M. Hajra, and M. Ghosh. ”Supervised classification algorithms in machine learning: A survey and review”. in ***Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph*** *2018*. 2020. Springer.

51. Nodet, P., et al. ”From weakly supervised learning to biquality learning: an introduction”. in *2021* ***International Joint Conference on Neural Networks (IJCNN)***. 2021. IEEE.

52. Yang, L., et al., ”Task offloading for directed acyclic graph applications based on edge computing in industrial internet”. ***Information Sciences***, 2020. **540**: p. 51-68.

53. Mahesh, B., ”Machine learning algorithms-a review”**.** ***International Journal of Science and Research (IJSR)***.[Internet], 2020. **9**: p. 381-386.

54. Bonaccorso, G., ”Machine learning algorithms”. 15-30 (2017): ***Packt Publishing Ltd***.

55. Mohammed, M., M.B. Khan, and E.B.M. Bashier, ”Machine learning: algorithms and applications”.45-68 (2016): ***Crc Press.***

56. Ayodele, T.O., ”Types of machine learning algorithms”. ***New advances in machine learning***, 2010. **3**: p. 19-48.

57. Mitchell, T.M., "Machine learning". 12-42 (Vol. 1. 2007): ***McGraw-hill New York.***

58. Vabalas, A., et al., ”Machine learning algorithm validation with a limited sample size”. ***PloS one***, 2019. **14**(11): p. e0224365.

59. Fix, E., ”Discriminatory analysis: nonparametric discrimination, consistency properties”. Vol. 1. 1985: ***USAF school of Aviation Medicine***.

60. Cover, T. and P. Hart, ”Nearest neighbor pattern classification”. ***IEEE transactions on information theory***, 1967. **13**(1): p. 21-27.

61. Zhang, S., et al., ”Learning k for knn classification”. ***ACM Transactions on Intelligent Systems and Technology (TIST)***, 2017. **8**(3): p. 1-19.

62. Deng, Z., et al., ”Efficient kNN classification algorithm for big data”. ***Neurocomputing***, 2016. **195**: p. 143-148.

63. Xing, W. and Y. Bei, ”Medical health big data classification based on KNN classification algorithm”. ***IEEE Access***, 2019. **8**: p. 28808-28819.

64. Hastie, T., et al., ”The elements of statistical learning: data mining, inference, and prediction”. Vol. 2. 2009: ***Springer***.

65. Jaskowiak, P.A. and R. Campello. ”Comparing correlation coefficients as dissimilarity measures for cancer classification in gene expression data”. in ***Proceedings of the Brazilian symposium on bioinformatics***. 2011. Brasília Brazil.

66. Zhang, S., et al., ”A novel kNN algorithm with data-driven k parameter computation”. ***Pattern Recognition Letters***, 2018. **109**: p. 44-54.

67. Bonchev, D. and N. Trinajstić, ”Information theory, distance matrix, and molecular branching”. ***The Journal of Chemical Physics***, 1977. **67**(10): p. 4517-4533.

68. Lele, S. and J.T. Richtsmeier, ”Euclidean distance matrix analysis: confidence intervals for form and growth differences”. ***American journal of physical anthropology***, 1995. **98**(1): p. 73-86.

69. Bakonyi, M. and C.R. Johnson, ”The Euclidian distance matrix completion problem”. ***SIAM Journal on Matrix Analysis and Applications***, 1995. **16**(2): p. 646-654.

70. Weyenberg, G. and R. Yoshida, ”Reconstructing the phylogeny: Computational methods, in Algebraic and Discrete Mathematical methods for modern Biology”. 2015, ***Elsevier***. p. 293-319.

71. Norman, R.Z., ”Structural models: An introduction to the theory of directed graphs”. 1965. ***Wiley*** 1965.

72. Graham, R.L., A.J. Hoffman, and H. Hosoya, ”On the distance matrix of a directed graph”. ***Journal of Graph Theory***, 1977. **1**(1): p. 85-88.

73. Babić, D., et al., ”Resistance-distance matrix: a computational algorithm and its application”. ***International Journal of Quantum Chemistry***, 2002. **90**(1): p. 166-176.

74. Nazeer, K.A. and M. Sebastian. ”Improving the Accuracy and Efficiency of the k-means Clustering Algorithm”. in ***Proceedings of the world congress on engineering***. 2009. Association of Engineers London London, UK.

75. Hamerly, G. and C. Elkan, ”Learning the k in k-means”. ***Advances in neural information processing systems***, 2003. **16**.

76. Pelleg, D. and A. Moore. ”Accelerating exact k-means algorithms with geometric reasoning”. in ***Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining***. 1999.

77. Yu, S.-S., et al., ”Two improved k-means algorithms”. ***Applied Soft Computing***, 2018. **68**: p. 747-755.

78. Yedla, M., S.R. Pathakota, and T. Srinivasa, ”Enhancing K-means clustering algorithm with improved initial center”. ***International Journal of computer science and information technologies***, 2010. **1**(2): p. 121-125.

79. Sing, J., et al. ”Improved k-means algorithm in the design of RBF neural networks”. in ***TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region***. 2003. IEEE.

80. internet: Maheswaran, G., et al., ”K Means Clustering Algorithms: A Comparitive Study”. **https://people.cse.nitc.ac.in/sites/default/files/govind/files/report.pdf**

81. internet: Bayes, N., ”K-means clustering”. 2015. **https://debategraph.org/Details.aspx?nid=305161**

82. Lye, K.O., S. Mishra, and R. Molinaro, ”A multi-level procedure for enhancing accuracy of machine learning algorithms”. ***European Journal of Applied Mathematics***, 2021. **32**(3): p. 436-469.

83. Sridhar, K., et al., ”Enhanced Machine learning algorithms Lightweight Ensemble Classification of Normal versus Leukemic Cel”. ***Journal of Pharmaceutical Negative Results***, 2022: p. 496-505.

84. Ahsan, M., et al., ”Enhancing machine learning prediction in cybersecurity using dynamic feature selector”. ***Journal of Cybersecurity and Privacy***, 2021. **1**(1): p. 199-218.

85. Friedler, S.A., et al. ”A comparative study of fairness-enhancing interventions in machine learning”. in ***Proceedings of the conference on fairness, accountability, and transparency***. 2019.

86. Sbarufatti, C., G. Manson, and K. Worden, ”A numerically-enhanced machine learning approach to damage diagnosis using a Lamb wave sensing network”. ***Journal of Sound and Vibration***, 2014. **333**(19): p. 4499-4525.

87. Comar, P.M., et al. ”Combining supervised and unsupervised learning for zero-day malware detection”. in *2013* ***Proceedings IEEE INFOCOM***. 2013. IEEE.

88. Corsini, P., B. Lazzerini, and F. Marcelloni, ”Combining supervised and unsupervised learning for data clustering”. ***Neural Computing & Applications***, 2006. **15**: p. 289-297.

89. Sedaghat, N., et al., ”Combining supervised and unsupervised learning for improved miRNA target prediction”. ***IEEE/ACM transactions on computational biology and bioinformatics***, 2017. **15**(5): p. 1594-1604.

90. Omta, W.A., et al., ”Combining supervised and unsupervised machine learning methods for phenotypic functional genomics screening”. ***SLAS Discovery***, 2020. **25**(6): p. 655-664.

91. internet: knowbe4, ”Phishing. 2023”. **https://blog.knowbe4.com/q1-2023-top-clicked-phishing.**

92. Wang, J., et al., ”An exploration of the design features of phishing attacks”. ***Information Assurance, Security and Privacy Services***, 2009. **4**(29): p. 178-199.

93. Alghenaim, M.F., N.A.A. Bakar, and F.A. Rahim. ”Awareness of Phishing Attacks in the Public Sector: Review Types and Technical Approaches”. in ***Proceedings of the 2nd International Conference on Emerging Technologies and Intelligent Systems: ICETIS*** *2022 Volume 1*. 2023. Springer.

94. bin Othman Mustafa, M.S., et al. ”An enhanced model for increasing awareness of vocational students against phishing attacks”. in ***2019 IEEE international conference on automatic control and intelligent systems (I2CACIS)***. 2019. IEEE.

95. Chandrasekaran, M., K. Narayanan, and S. Upadhyaya. ”Phishing email detection based on structural properties”. in ***NYS cyber security conference***. 2006. Albany, New York.

96. Tan, C.L., ”Phishing dataset for machine learning: Feature evaluation”. ***Mendeley*** Data, 2018. **1**: p. 2018.

97. internet: TIWARI, T. ”Phishing Site URLs”. 2019; Available from: **https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls**.

98. Internet: Dileep, K. ”Heart Disease Prediction using Logistic Regression”. 2023 [cited 2023 April 1, 2023]; Available from: **https://www.kaggle.com/datasets/dileep070/heart-disease-prediction-using-logistic-regression**.

99. Internet: Repository, U.M.L. ”Indian Liver Patient Records”. 2019 [cited 2023 April 1, 2023]; Available from: **https://www.kaggle.com/datasets/uciml/indian-liver-patient-records.**

100. Internet: Sulianova, A. ”Cardiovascular Disease Dataset”. 2023 [cited 2023 April 1, 2023]; Available from: **https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset.**

101. Haghighi, S., et al., ”PyCM: Multiclass confusion matrix library in Python”. ***Journal of Open Source Software***, 2018. **3**(25): p. 729.

102. Deng, X., et al., ”An improved method to construct basic probability assignment based on the confusion matrix for classification problem”. ***Information Sciences***, 2016. **340**: p. 250-261.

103. Room, C., ”Confusion Matrix”. ***Mach. Learn***, 2019. **6**: p. 27.

104. Chicco, D., N. Tötsch, and G. Jurman, ”The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation”. ***BioData mining***, 2021. **14**(1): p. 1-22.

105. Huang, Z., X. Li, and H. Chen. ”Link prediction approach to collaborative filtering”. in ***Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries***. 2005.

106. Ozdemir, S. and D. Susarla, ”Feature Engineering Made Easy: Identify unique features from your dataset in order to build powerful machine learning systems”. 2018: ***Packt Publishing Ltd***.

107. Zheng, A. and A. Casari, ”Feature engineering for machine learning: principles and techniques for data scientists”. 2018: ***" O'Reilly Media, Inc***.".

108. Ng, A., *Machine Learning and AI via Brain simulations.* Accessed: May, 2013. **3**: p. 2018.

109. Alarbi, A. and Z. Albayrak, ”Core Classifier Algorithm: A Hybrid Classification Algorithm Based on Class Core and Clustering”. ***Applied Sciences***, 2022. **12**(7): p. 3524.

RESUME

Abdalraouf Almahdi Mohammed ALARBI he received his first and elementary education. He finished high school at Soukna High School and then earned his undergraduate degree from The Higher Institute for Comprehensive Professions in Soukna in 2002. Then, in 2011, he began his Master's degree at the University of Bridgeport in the United States, where he graduated in Fall 2013. In 2017, he transferred to Karabük University to begin his Ph.D. studies in the Department of Computer Engineering.