A Machine Learning Approach to Identifying Malicious DNS Requests through Server Log Analysis

Dissertation submitted in partial fulfilment of the requirement
for the degree of

Master of Science in Data Analytics

at Dublin Business School

10623040

Supervisor: Satya Prakash

August 2023

Declaration

I hereby declare that the dissertation presented to Dublin Business School for the purpose of obtaining the [MSc. Data Analytics] degree is the culmination of my independent research endeavors, except in instances explicitly acknowledged through proper citations. Additionally, I confirm that this work has not been presented for any other academic qualification.

Signed: Teja Kolla

Student Number: 10623040

Date: August 2023

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Satya Prakash, for his invaluable guidance and support throughout this research.

I am also thankful to my family and friends for their unwavering encouragement and belief in my abilities.

This work would not have been possible without the resources provided by Dublin Business School

Lastly, I extend my appreciation to all participants and contributors who played a role in this research.

Teja Kolla

[August,2023]

Abstract

In the dynamic landscape of digital connectivity, the Domain Name System (DNS) plays a significant role in internet infrastructure, enabling the translation of human readable domain names into machine understandable IP addresses. Unfortunately, this critical service also presents a vulnerable entry point for cyber attackers to execute a range of malicious activities including phishing, malware distribution, and domain hijacking. Traditional manual analysis of DNS traffic struggles to cope with the volume and complexity of modern cyber threats. To address this challenge, a comprehensive approach is proposed that harnesses the capabilities of machine learning for the identification of malicious DNS requests through server log analysis. The primary objective of this research is to design, implement, and evaluate a robust machine learning framework capable of distinguishing between benign and malicious DNS requests. Using a diverse dataset of server logs, appropriate preprocessing techniques are employed to cleanse and transform the raw data into a suitable format for analysis. The approach focuses on the identification of relevant features and the engineering of domain-specific attributes that capture the behavior of both legitimate and malicious requests. Through a comprehensive evaluation process, a range of machine learning algorithms suitable for classification tasks are explored. The chosen models undergo critical assessment using established evaluation metrics to quantify their performance in differentiating between malicious and benign DNS requests.

Keywords: Domain Name System (DNS), machine learning, malicious requests, server log analysis, cybersecurity, classification algorithms.

Contents

Table of Figures

Chapter 1: Introduction

In today's digital landscape, the rapid growth of online activities has led to an increase in cyber threats and attacks. One of the prominent attack vectors involves the Domain Name System (DNS), which is a fundamental component of the internet infrastructure responsible for translating human-readable domain names into IP addresses. Malicious actors exploit DNS to carry out various cyber-attacks, including domain hijacking, phishing attacks, malware distribution, and command-and-control communication

Detecting these malicious activities in DNS requests is of paramount importance to ensure the security and integrity of online services and networks. Traditional methods of manual analysis fall short due to the sheer volume of data generated by DNS requests and the evolving tactics of cyber attackers.

In the digital realm, where communication occurs between devices through numerical IP addresses, DNS acts as an essential intermediary, facilitating easy access to websites and online resources. This hierarchical system functions much like a phone directory for the internet, enabling users to navigate the vast online landscape using familiar domain names, such as www.example.com, instead of numerical IP addresses.

At its core, DNS operates through a series of servers that collaboratively manage the translation process. When a user enters a domain name in their web browser, the DNS system sequentially queries a hierarchy of servers, starting from local caches to authoritative name servers, until it obtains the corresponding IP address. This process is crucial for enabling seamless connectivity across the internet, allowing users to effortlessly access websites, send emails, and engage in various online activities.

DNS serves as the backbone of the internet's addressing system, ensuring that users can navigate the digital landscape intuitively and efficiently. Given its indispensable role, any disruption or compromise to the DNS system can have far-reaching implications for online services, security, and privacy. Therefore, understanding and securing the DNS ecosystem against potential threats and attacks is of paramount importance to maintaining a safe and resilient online environment.

Web servers play a critical role in the digital economy, providing a platform for businesses to interact with customers and conduct transactions. However, with this increased reliance on web servers comes an increased risk of cyber-attacks, with attackers seeking to exploit vulnerabilities in web applications to gain access to sensitive data or disrupt operations. Detection of anomalies in web traffic is an essential component for a web application for providing insights into potential threats and vulnerabilities.

Traditionally, prevention of attacks in web traffic has relied on rule-based methods to prevent attacks on a web server. These methods are often limited by their static nature and lack of scalability. With the growing volume, complexity and dynamic nature of web attacks, there is a need for more sophisticated approaches that can handle large attacks and adapt to changing threats. This is where the integration of machine learning techniques becomes essential.

Many intrusion detection systems have been developed, however most of these systems focus on analysis of packets transferred at the transport layer. In this proposed research, the aim is to develop a machine learning-based approach for anomaly detection in web server logs, which is at application layer. The primary objective of this research is to investigate the effectiveness of machine learning algorithms for detecting anomalies and pattern recognition in web server logs and build a classifier model that can improve the accuracy and robustness of the detection system by evaluating the predictions of multiple classifiers.

1.1 Problem Statement:

The core challenge this research endeavors to address is the identification and classification of malicious DNS requests within a large amount of legitimate DNS traffic. Manual analysis and rule-based approaches, while once effective, are now struggling to contend with the subtleties of modern cyber-attack strategies, which often employ complex techniques to evade detection. Consequently, a more proactive, automated, and scalable approach is required to accurately differentiate between benign and malicious DNS requests.

1.2 Research Objectives:

The primary objective of this research is to design, implement, and evaluate a robust machine learning framework capable of identifying malicious DNS requests based on patterns, behaviors, and attributes extracted from server log data. The framework aims to empower organizations with an advanced tool to detect and counteract cyber threats proactively, enhancing their overall cybersecurity posture.

1.3 Research Questions:

This research seeks to address the following key questions:

1. How can machine learning techniques be harnessed to effectively identify and classify malicious DNS requests from server log data?
2. What are the most relevant features and attributes that enable accurate differentiation between legitimate and malicious DNS requests?
3. How does the performance of various machine learning algorithms compare in the context of DNS request classification?

1.4 Significance and Motivation:

The significance of this research extends beyond conventional cybersecurity practices that often focus on analyzing network and transport layer activities. In many instances, the detection of anomalies and malicious activities involves scrutinizing packet-level information, a methodology that has proven effective at identifying external threats. However, this approach falls short in identifying threats and attacks that operate at the application layer—where requests and responses are processed by web applications. This is where our research assumes paramount importance.

By concentrating on the application layer and leveraging the vast potential of machine learning, this research explores a novel avenue for identifying and categorizing malicious DNS requests. Unlike

traditional methods that often rely on dissecting packets, our approach harnesses the wealth of information stored within server logs. The application layer is a treasure trove of insights into user behavior, requests, and interactions, making it an invaluable source for uncovering nuanced threats that might evade detection at lower layers of the network stack.

Moreover, the application layer's emphasis on user interactions renders it more closely aligned with the goals of modern cyber attackers, who seek to exploit vulnerabilities in web applications for financial gain, data theft, and disruption of services. By fortifying the application layer's defenses, our research aligns with the evolving threat landscape and provides organizations with a proactive means of countering sophisticated cyber threats.

In summary, this research is motivated by the imperative to expand the realm of cybersecurity beyond traditional paradigms. By shifting the focus to the application layer and leveraging the capabilities of machine learning, proposed approach offers a new perspective on identifying malicious DNS requests—adding a potent tool to the arsenal of defenders against the ever-evolving tactics of cyber attackers.

1.5 Scope and Limitations:

The scope of this research is centered on the identification of malicious DNS requests through the lens of machine learning and server log analysis. The research methodology encapsulates the collection, preprocessing, and analysis of DNS request data, leading to the development and evaluation of machine learning models capable of distinguishing between benign and malicious requests. The objective is to enhance the cybersecurity framework by addressing a specific yet critical aspect of cyber threats.

However, it is important to acknowledge that server logs have a wealth of information beyond DNS requests alone. The server logs encompass a comprehensive record of user behavior, HTTP requests, potential SQL injection attacks, and many other ‘Application layer’ activities. While this research maintains a specific focus on DNS requests, the methodology and principles can indeed be extended to encompass a broader spectrum of security threats.

The limitations of this research stem from its targeted scope. By concentrating solely on DNS requests, we inevitably omit the multifaceted landscape of application layer security challenges. While DNS request identification forms a crucial layer of defense against certain types of cyber threats, it is not a solution for all security vulnerabilities. Such as threats that operate at the application layer, including SQL injection attacks and sophisticated user behavior anomalies, warrant separate and dedicated research applications.

In summary, while this research focuses on a distinctive approach to DNS request analysis, it is vital to recognize that the potential of server log analysis extends beyond DNS requests. Future research endeavors can explore the broader realm of application layer security, encompassing diverse threats and attack vectors that demand specialized attention.

1.6 Outline:

The subsequent chapters of this dissertation will delve into the research methodology, data collection, preprocessing, exploratory data analysis, feature selection, model selection, evaluation, and the interpretation of results. The final chapters will provide conclusions, insights, and recommendations for future research.

In summary, this introductory chapter lays the foundation for a comprehensive exploration of the utilization of machine learning in the identification of malicious DNS requests. The subsequent chapters will delve into the intricacies of the research process, culminating in a holistic understanding of the potential and limitations of this groundbreaking approach to cybersecurity.

Chapter 2: Literature Review

2.1 Introduction: Pattern Recognition and Anomaly detection in server logs using machine learning techniques has been extensively studied. There are a wide range of algorithms and techniques proposed on server log analysis for various applications. In this section, we will review the literature on anomaly detection using machine learning techniques, identify the strengths and weaknesses of existing approaches, and highlight areas for future research.

2.2 Concepts of DNS and Malicious Requests:

The Domain Name System (DNS) stands as a cornerstone of modern internet functionality, translating human-readable domain names into numerical IP addresses that underpin online communication. This hierarchical, distributed system involves recursive and iterative queries across a network of servers to efficiently resolve domain names. DNS plays an integral role in enabling seamless navigation across the internet, underpinning services ranging from web browsing to email communication.

However, alongside the utility of DNS lies a vulnerability exploited by malicious actors. As stated in this study by (Rajendran, 2018) Malicious DNS requests encompass a spectrum of activities, from domain hijacking to DNS cache poisoning and distributed denial-of-service (DDoS) attacks. These requests aim to manipulate DNS responses, redirect users to malicious websites, or compromise the integrity of data transmission. Detecting and thwarting such malicious requests is imperative to ensuring the security and stability of online services.

Understanding malicious DNS requests necessitates a grasp of the intricacies of DNS protocol vulnerabilities, such as DNS amplification and tunneling. Moreover, the ability to distinguish between benign and malicious requests rests on analyzing patterns, query types, and behaviors that deviate from established norms. As a result, the study of malicious DNS requests involves not only a mastery of DNS functionality but also a nuanced comprehension of potential threat vectors and their characteristics.

Against this backdrop of vulnerabilities and historical attacks, two significant security advancements emerge: Domain Name System Security Extensions (DNSSEC) and DNS firewall solutions.

DNSSEC: Outlined in RFC 4033 (Rose, 2005), employs cryptographic digital signatures for data origin authentication and integrity enhancement. Although DNSSEC is widespread, only a minute percentage of domains implement it. While it thwarts spoofing and man-in-the-middle attacks, complexities in implementation and misconfigurations introduce vulnerabilities.

The subsequent sections dive into the exploration of DNS and malicious requests, and research efforts, insights that underpin the development of effective detection and prevention strategies.

DNS firewall: These solutions confront challenges posed by Domain Generation Algorithm (DGA)-based malicious activities (Plohmann, 2016). These algorithms, rooted in arithmetic, hash, wordlist, and permutation methods, create randomized domains, posing challenges for detection.

2.3 Previous Research in DNS Request Analysis and attack detection:

In this published study conducted by (Jin, 2019) the proposed method constructs a DNS database from cached DNS data history logs. The analysis phase extracts features from original DNS query-response pairs, maps them to corresponding cached DNS records, and populates the DNS database. Incoming DNS query-response pairs are analyzed in collaboration with the DNS database to determine whether to add received DNS records to the cache. False positives lead to identification and investigation of end clients based on historical DNS database logs. This study focuses on categorizing three specific attack patterns: legitimate cases, Kaminsky attacks, and DNS cache poisoning attacks from compromised authoritative DNS servers. Legitimate cases involve caching DNS records from legitimate responses.

The study done by (Ming Li, 2021) recognizes that APT attacks involve the compromise of specific hosts within a network, leading to unique behavioral patterns. One notable contribution is the identification of characteristic patterns in DNS request sequences made by compromised hosts over time. These patterns serve as distinctive features for detecting compromised hosts, even in the absence of known malicious domain samples. By focusing on temporal patterns within DNS logs, the study introduces an innovative approach to APT detection.

Additionally, the research contributes by conducting extensive evaluations on a real large-scale network environment, utilizing 70 days' worth of DNS request records. This practical validation demonstrates the effectiveness of their proposed approach in detecting APT compromises. Furthermore, the study validates its method using a public dataset, allowing for comparisons with existing detection methods.

This paper (Shalaginov, 2016) discusses the challenges and complexities associated with detecting malware beaconing through DNS logs, emphasizing the importance of periodicity detection. Malicious software often communicates with Command and Control (C2) servers by sending DNS queries for C2 server IP addresses through internal or external DNS servers. These DNS responses may contain encoded data, making it challenging to detect malware job-scheduling.

- Unpredictable sleep times: Malware activation and beaconing times are not known in advance.
- Multiple period usage: Attackers may change beaconing intervals over time.
- Time variation: Attackers vary sleep times to avoid detection, mimicking legitimate communication patterns.
- Noise: Benign applications, such as system updates, can produce regular beacons.
- Multiple channels usage: Malware may shift between different C2 servers.
- Benign beacon: Some legitimate applications generate regular beacons.
- Needle in a haystack: Large enterprises generate massive log volumes, making it challenging to identify malicious activities quickly.
- Near real-time detection: Identifying malicious beacon events as soon as possible is crucial, but it can be challenging due to the sheer volume of network traffic and the delay between malware launch and beaconing.

This paper (Yan, 2020) introduces a novel system for Advanced Persistent Threat (APT) attack detection through DNS logs. Thorough log analysis yielded seven DNS features strongly correlated with suspicious APT activities. Leveraging recent advances in machine learning, a neural network model was developed

to find out the detailed relationship between DNS behavior and APT attacks. The model achieved compelling results, boasting a 96.8% recall and a 97.6% accuracy. The contributions of this work are manifold. Identification of seven DNS features spanning three categories and eight DNS features connected to APT activities. Introduction of three novel DNS behavior-related features, specifically centered on the interaction between DNS request and response messages, as well as the temporal attributes of DNS logs. Development of a robust system with the help of deep learning for detecting malicious DNS behavior within APT attacks and generating a feature set.

One of the most common approaches to anomaly detection using machine learning is based on clustering algorithms one such technique is proposed in (Juvonen, 2012) . Clustering algorithms group data points together based on their similarity, and anomalies are identified as data points that do not belong to any of the clusters. K-means and DBSCAN are two examples of clustering algorithms that have been used for anomaly detection in various applications. The strengths of clustering-based anomaly detection include their ability to identify previously unknown anomalies and their ability to handle large datasets. However, clustering algorithms can be sensitive to the choice of hyper-parameters and may require manual tuning, which can be time-consuming and require expert knowledge.

Another approach to anomaly detection using machine learning is based on classification algorithms. Classification algorithms learn a model based on labelled data and use this model to classify new data points as either normal or anomalous. Decision trees, support vector machines, and Random Forest are examples of classification algorithms that have been used for anomaly detection. There is comparison study done by using different classification algorithms in (Pham, 2016). The strengths of classification-based anomaly detection include their ability to handle high-dimensional data and their ability to detect anomalies in real-time. However, classification algorithms require labeled data for training, which can be difficult to obtain in some applications. Moreover, classification algorithms may struggle to identify previously unknown anomalies that do not fit into the learned model.

A third approach to anomaly detection using machine learning is based on Classification using deep learning algorithms such as CNNs and RNNs. DeepLog (Du, 2017) is one such model built on deep learning algorithm. However deep learning models need training on large amount of normal data (non-anomalous server logs) in order to identify anomalies in new logs. Deep learning based algorithms offer several advantages in terms of their ability to handle high-dimensional data, identify anomalies, and handle noisy data, this is evident in the study done in (Zhang, 2022).

Clustering-based, classification-based, and deep learning based approaches all have their strengths and weaknesses, and the choice of approach depends on the specific requirements of the application. Future research should focus on developing more efficient and scalable algorithms, addressing the challenge of detecting previously unknown anomalies, and exploring the use of ensemble methods for anomaly detection that can effectively handle large-scale web server logs.

2.4 Challenges and Gaps in the Literature:

A notable gap and challenge is the limited focus on real-time anomaly detection in the context of DNS log analysis. While various studies discuss the identification of APTs and malicious behavior through DNS logs,

many of these methods rely on historical data or batch processing, which may not be suitable for timely threat mitigation.

Addressing this gap is essential because advanced threats often require rapid responses to prevent significant damage. Developing real-time anomaly detection techniques, potentially incorporating Machine-Learning models such as CNNs and RNNs, could significantly enhance network security by identifying suspicious DNS behaviors as they occur. However, this approach may present challenges related to processing speed, scalability, and handling noisy data in real-time scenarios.

Future research should aim to bridge this gap by exploring and developing effective real-time DNS anomaly detection methods, considering the challenges of handling the high data volume and ensuring timely threat identification and response. This would contribute to strengthening network security against APTs and other emerging threats.

Chapter 3: Research Methodology

3.1 Research Design and Approach:

In this research, a systematic approach is employed to investigate and address the complex challenges associated with "Identifying malicious DNS requests from server log analysis using machine learning." The research methodology follows a well-established CRISP-DM (Cross-Industry Standard Process for Data Mining) framework originally proposed by (Shearer, Fall 2000), which provides a structured and comprehensive guideline for conducting data-driven research and analysis for data mining.

CRISP-DM emerges as an ideal choice for this investigation due to its versatility and applicability across diverse domains. It furnishes a systematic roadmap for every phase of the study, spanning from data collection to model evaluation, effectively guiding the exploration of the multifaceted domain of malicious DNS request detection.

By adhering to the CRISP-DM framework, this research ensures methodological rigor and the potential for replication. It promotes efficient communication and synergy among research team members, facilitating the harnessing of collective expertise to tackle the multifarious dimensions of malicious DNS request identification.

This approach seamlessly aligns with the research objectives of unveiling hidden patterns within server logs, analyzing them with machine learning techniques, and subsequently identifying malicious DNS requests effectively. Through systematic adherence to the CRISP-DM framework, this research aims to contribute significantly to enhancing our comprehension of malicious DNS request detection and fortifying cybersecurity measures.

3.2 Data Collection (Server Log Analysis):

3.2.1 Data Source and Description:

The dataset used in this research originates from publicly available DNS logs published by (Magalhães, 2021), encompassing both malicious and non-malicious domain names. It was meticulously curated and constructed from scratch to serve as a robust foundation for the application of supervised machine learning techniques in the classification of malicious and non-malicious domain names. To generate this dataset, comprehensive features were extracted from the domain names, including metrics like domain name entropy, length, and the presence of unusual characters. In addition, various other attributes such as domain creation date, Internet Protocol (IP) details, open ports, and geolocation information were acquired through data enrichment processes, specifically Open Source Intelligence (OSINT).

The categorization of domain names into their respective classes was determined based on the data source, distinguishing between those sourced from malicious DNS log files and those originating from non-malicious DNS log files. Notably, this dataset demonstrates a balanced composition, with approximately 90,000 domain names evenly distributed, comprising 50% non-malicious and 50% malicious domain names. This dataset stands as a valuable resource for the application of machine

learning algorithms, enabling the development of models capable of discerning between malicious and non-malicious domain names with enhanced accuracy and efficiency.

3.3 Data Preprocessing:

Data preprocessing and Transformation stages are essential for cleaning, structuring, and enhancing the raw data, ensuring that data is in a suitable form for model building and analysis. In this section, we will explore these data preprocessing and transformation steps in detail, providing comprehensive insights in to the data.

3.3.1 Handling Missing Values:

Missing data is a common issue encountered when working with real-world datasets. In the dataset, missing values were observed in specific columns such as 'CountryCode,' 'RegisteredCountry,' and 'RegisteredOrg.' Dealing with missing values is crucial, as they can lead to biased results and hinder the performance of machine learning models. In this context, a pragmatic approach was adopted: the columns containing missing values were removed. This decision was made because these columns were deemed non-critical for the research question, and the missing data in these columns did not provide substantial value for the analysis.

Moreover, a small percentage of rows in the dataset contained missing values. While this percentage was relatively minor, it's essential to address missing values to maintain data integrity. Therefore, these incomplete records were removed from the dataset. By doing so, data consistency and quality were upheld, reducing the potential for errors in subsequent analyses. Eliminating rows with missing values is a standard practice when the proportion of missing data is minimal and does not significantly impact the dataset's overall quality or size.

3.3.2 Label Encoding:

Machine learning algorithms primarily operate with numerical data. However, the dataset initially included non-numerical data, such as categorical features (columns with string data type). To facilitate the use of these features in machine learning models, a label encoding technique was employed. Label encoding involves assigning a unique numerical label to each category or class within a categorical feature. This transformation ensures that the algorithms can effectively process and learn from the data.

3.3.3 Min-Max Normalization:

After encoding the dataset into numerical values, it was essential to address potential discrepancies in the scales of different features. Features with significantly different scales can negatively impact the performance of machine learning algorithms, particularly distance-based models. To mitigate this issue, min-max normalization was applied. This technique scales features to a specific range, typically between 0 and 1. Normalizing the data in this manner ensures that all features have a consistent scale, promoting fair treatment of each feature during model training.

3.3.4 Standard Scaling:

Another normalization technique applied in the data preprocessing pipeline was standard scaling, also known as Z-score normalization. Standard scaling transforms features to have a mean of 0 and a standard deviation of 1. This process standardizes the data and centers it around zero. Standard scaling is particularly useful when working with algorithms that are sensitive to feature scales. By standardizing the data, you ensure that all features are treated equally in terms of their distribution, preventing any single feature from dominating the modeling process.

3.3.5 Principal Component Analysis (PCA):

One of the challenges when dealing with datasets containing a large number of features is high dimensionality. High dimensionality can lead to increased computational complexity and may even result in overfitting, where a model performs well on training data but poorly on unseen data. To address this challenge, Principal Component Analysis (PCA) was employed. PCA is a dimensionality reduction technique that projects the original dataset into a lower-dimensional space while preserving as much variance as possible. It achieves this by identifying the principal components of the data, which are linear combinations of the original features. These principal components capture the most critical information in the data while reducing its dimensionality.

3.3.6 Summary:

In summary, data preprocessing and transformation are fundamental steps in the data mining and machine learning pipeline, ensuring that the dataset is appropriately prepared for analysis. Handling missing values, label encoding, normalization through techniques like min-max scaling and standard scaling, and dimensionality reduction using PCA are integral components of this process. By meticulously preparing and transforming the data, the foundation is set for more accurate and reliable machine learning models, ultimately enhancing the effectiveness for identifying malicious DNS requests from server log analysis using machine learning.

3.4 Exploratory Data Analysis:

3.4.1 Basic Statistics: Basic statistics serve as the foundation of EDA. It involves calculating fundamental statistical metrics for each feature in the dataset, such as mean, median, standard deviation, minimum, and maximum values. These statistics provide an initial understanding of the data's central tendencies, spread, and overall distribution. Basic statistics help identify potential outliers and assess data quality.

3.4.2 Univariate Analysis: Univariate analysis is the examination of individual features in isolation. It focuses on understanding the distribution, central tendencies, and variability of each feature. This analysis typically involves creating visualizations like histograms, box plots, and density plots to visualize the data's shape and characteristics. Univariate analysis helps identify outliers and assess the presence of skewed or non-normal distributions in the data.

Distribution of Target Variable: The dataset is evenly split between malicious and non-malicious categories, with an equal number of rows (45k) in each.

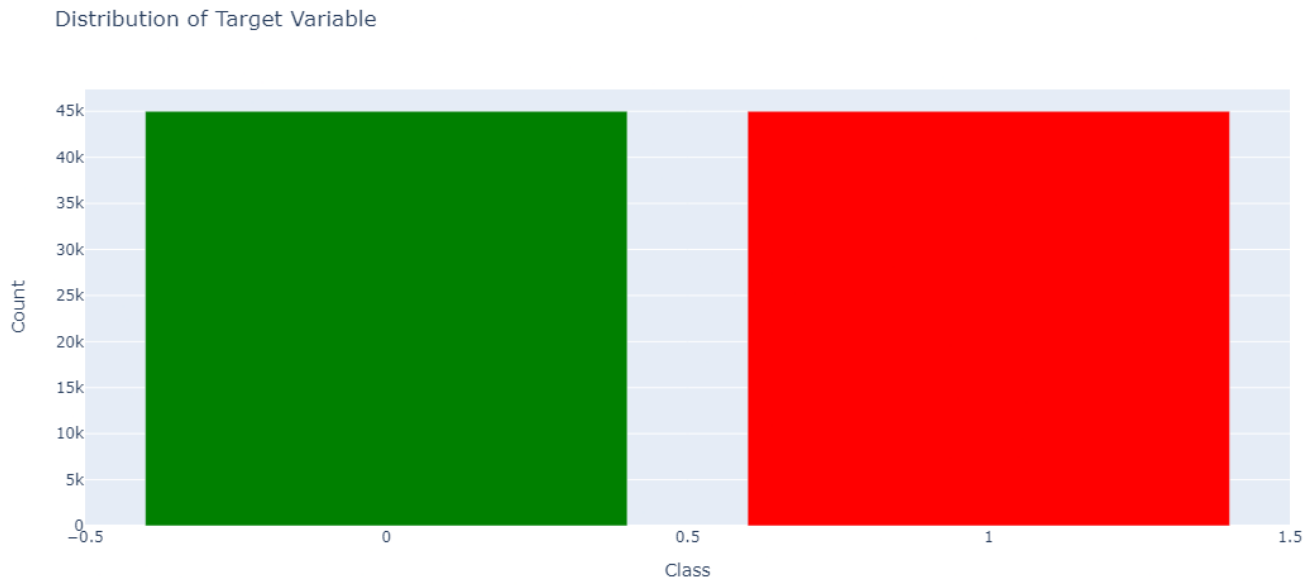


Figure 1 : Distribution of Target Variable

Distribution of DNSRecordType: There are very few 'MX' type records

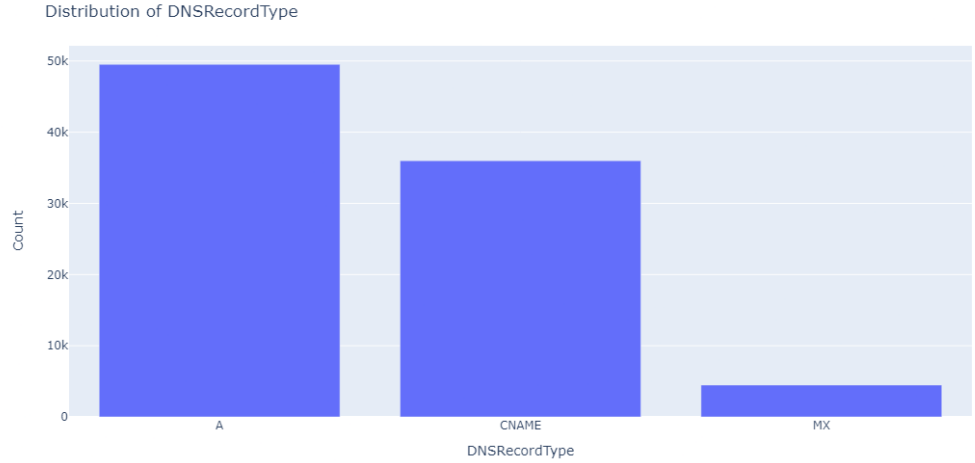


Figure 2: Distribution of DNSRecordType

3.4.3 Bivariate Analysis: Bivariate analysis explores relationships between pairs of variables, with a primary focus on understanding how the target variable interacts with other features. In your context, you mentioned visualizing relevant boolean columns with the target variable. This step is essential for measuring the skewness of the distribution concerning the target variable. Visualizations such as bar plots

or stacked bar plots can be used to display the distribution of boolean variables concerning the target. This helps assess how certain features may influence the likelihood of a DNS request being malicious.

Distribution of feature CommonPorts with Target variable

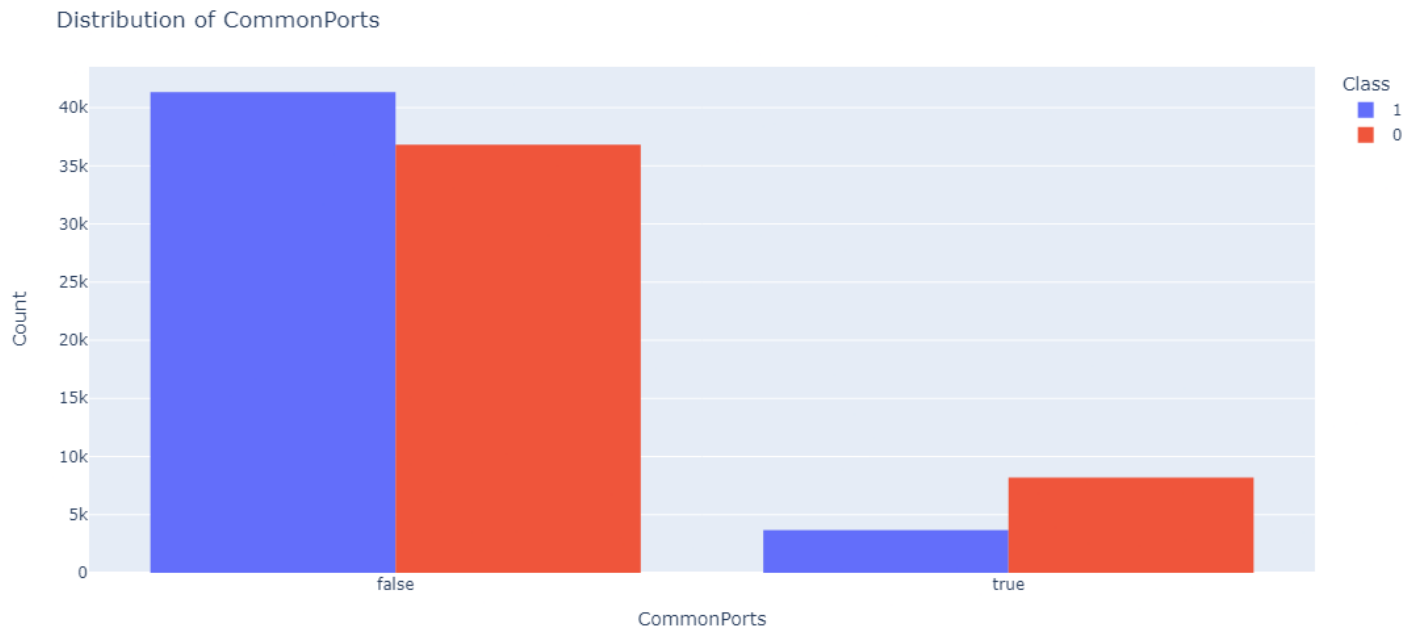


Figure 3 : Distribution of feature CommonPorts with Target variable

In the case of the 'CommonPorts' feature, the data distribution is notably biased towards the non-malicious category.

Distribution of feature TXTDnsResponse with Target variable

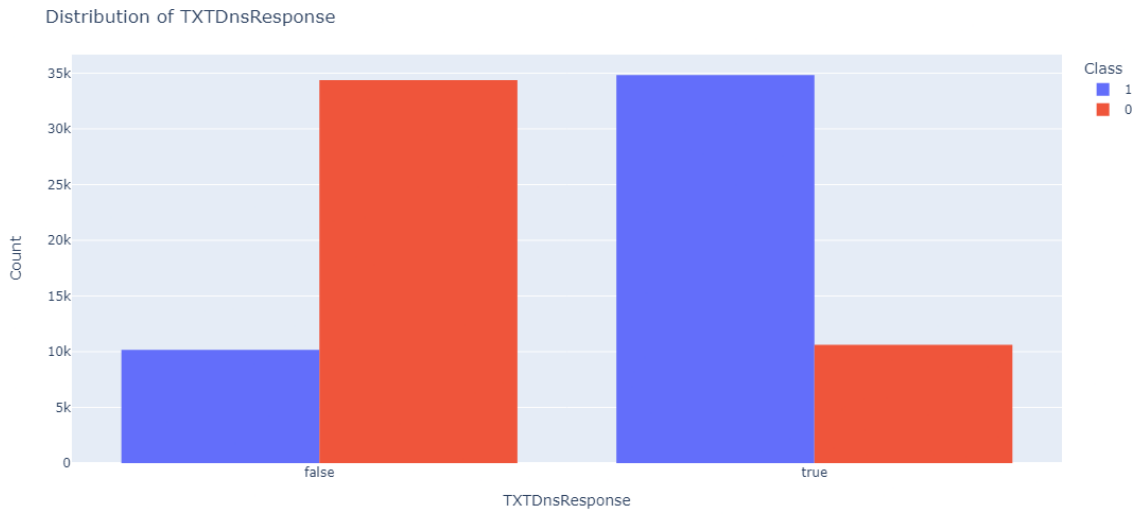


Figure 4: Distribution of feature TXTDnsResponse

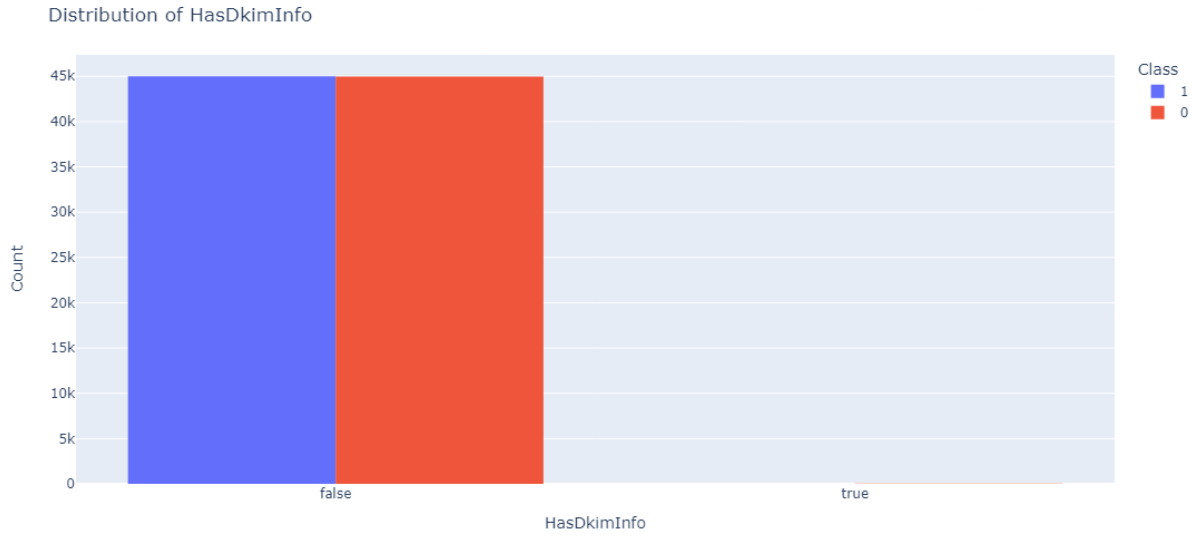


Figure 5: Distribution of feature HasDkimInfo with Target variable

In this particular feature 'HasDkimInfo' the data distribution is heavily skewed towards non-malicious category

3.4.4 Correlation Matrix: Building a correlation matrix is a vital step in understanding the relationships between numerical variables in your dataset. This matrix provides a numerical representation of the degree and direction of association between pairs of features. In your case, it can reveal whether there are any significant correlations between variables related to DNS requests and whether these correlations are positive or negative. A correlation matrix aids in identifying potential multi collinearity, which can impact the model interpretability.

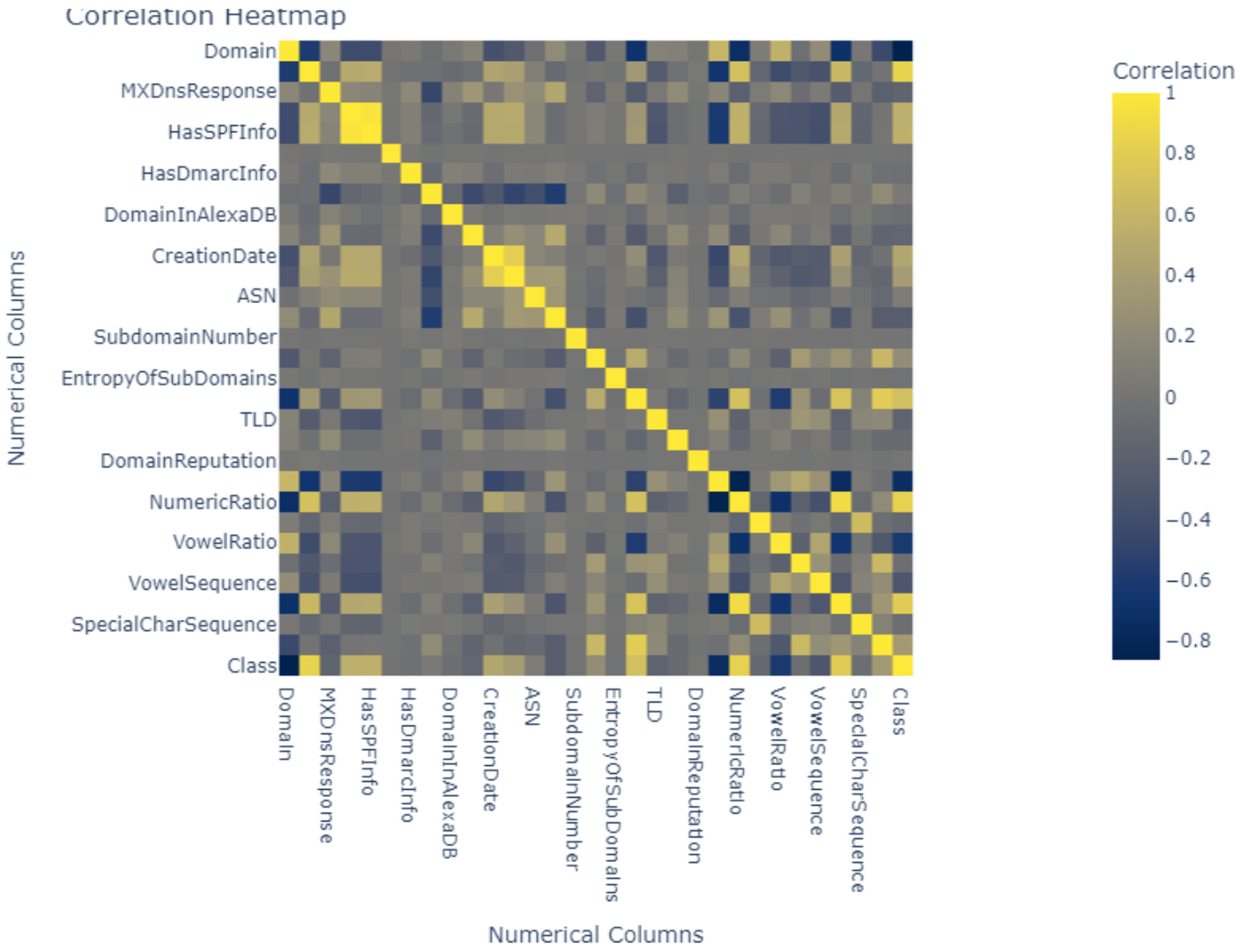


Figure 6: Correlation Heat map

From the correlation heat map, it is evident that some features exhibit strong correlations with each other, while others show significant correlations with the target variable.

3.5 Feature Selection and Engineering:

The primary research objective at hand is the precise identification of the most relevant features and attributes facilitating the accurate differentiation between legitimate and malicious DNS requests. In this section, we delve into the comprehensive process of feature selection and engineering, a pivotal step in crafting an effective model for identifying malicious DNS requests from server log analysis.

3.5.1 Feature Importance:

To prioritize features that exhibit the highest discriminative power between legitimate and malicious DNS requests, the Extra Trees Classifier algorithm is employed. This algorithm evaluates the importance of each feature and ranks them accordingly. From the large number of attributes, the top 20 features, contributing most significantly to the classification task, are selected. This step focuses the model's attention on the most informative aspects of DNS request data, thereby enhancing accuracy and efficiency.

3.5.2 Recursive Feature Elimination (RFE):

To extract the most important feature set to its most essential components, the Recursive Feature Elimination (RFE) technique is employed. RFE operates by iteratively fitting a logistic regression model and identifying the feature with the lowest importance. This feature is then removed from consideration, and the process is repeated until the desired feature count of 20 is achieved. RFE effectively sifts through the feature set, retaining only those attributes that contribute significantly to the model's predictive capacity. This eliminates irrelevant or redundant features, streamlining the model and reducing the risk of overfitting.

3.5.3 PCA Transformation:

Principal Component Analysis (PCA) offers a robust method for transforming the feature space from higher dimensions to a lower-dimensional representation. While the initial feature set is comprehensive, it may contain multicollinearity issues or excessive dimensionality that could hinder model performance. PCA addresses these concerns by identifying orthogonal axes, or principal components, along which the data exhibits the most significant variation. By retaining a selected number of these principal components, dimensionality is reduced without sacrificing crucial information. This transformation expedites computation and enhances the interpretability of the model.

The importance of feature selection and engineering cannot be overstated in the context of this research. The objective is to differentiate between benign and malicious DNS requests, ensuring that the model is trained on the most pertinent attributes. Selecting and refining these features significantly contributes to model efficiency, interpretability, and generalization.

In summary, the approach to feature selection and engineering encompasses multiple stages, each designed to enhance the discriminatory power of the model. By focusing on the top 20 features through Extra Trees Classifier and RFE, the attributes most relevant to the research objective are pin pointed. Additionally, PCA transformation enables navigation of the challenges of high-dimensional data, ensuring that the model is both effective and efficient. These steps collectively pave the way for a robust machine learning model capable of accurately identifying malicious DNS requests within server log data.

3.6 Data Modelling:

In the Data Modeling phase, the primary objective is to build and evaluate machine learning models that can effectively classify DNS requests as either benign or malicious. This phase typically involves several key steps:

3.6.1 Model Selection: The first step in Data Modeling is the selection of appropriate machine learning algorithms. In our case, we have explored a range of classification algorithms, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression, Random Forest, Decision Tree, Naive Bayes, and Gradient Boosting. Each algorithm has its unique characteristics, and their suitability depends on the dataset and problem at hand.

3.6.2 Data Splitting: To assess the performance of the selected models, the dataset is divided into two subsets: a training set and a testing set. The training set is used to train the models, while the testing set is used to evaluate their performance. Cross-validation techniques, such as k-fold cross-validation, has also been employed to ensure robust model assessment.

3.6.3 Model Training: During this step, the selected machine learning algorithms are trained on the training dataset. The models learn the underlying patterns and relationships within the data, enabling them to make predictions on unseen DNS requests.

3.6.4 Hyperparameter Tuning: Many machine learning algorithms have hyperparameters that control their behavior. Tuning these hyper parameters is a crucial part of optimizing model performance. Techniques like grid search or random search can be used to find the best combination of hyper parameters for each algorithm.

3.6.5 Model Evaluation: Once the models are trained, they are evaluated using the testing dataset. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics provide insights into how well the models can differentiate between benign and malicious DNS requests.

3.6.6 Ensemble Methods: Ensemble methods, such as bagging and boosting, may be employed to combine the predictions of multiple base models. Random Forest and Gradient Boosting are examples of ensemble algorithms that have been considered in this phase. Ensemble methods can often improve model performance by reducing overfitting and increasing predictive accuracy.

3.6.7 Model Selection: Based on the evaluation results, one or more models are selected as candidates for deployment. The choice of the final model(s) depends on the desired trade-offs between various performance metrics, computational resources, and interpretability.

3.6.8 Model Interpretation: In addition to selecting the best-performing models, it is essential to understand how these models arrive at their predictions. Interpretability techniques, such as feature importance analysis and visualization of decision boundaries, can provide insights into the factors that influence a DNS request's classification.

3.7 Classification algorithms

Below are the classification algorithms employed in this research, along with brief explanations of their operational principles and reasons for their applicability in classification tasks.

3.7.1 Random Forest:

Random Forest is an ensemble learning method that leverages the power of decision trees for classification. It combines multiple decision trees to make more robust and accurate predictions. In the context of identifying malicious DNS requests, Random Forest offers several advantages.

Random Forest operates by building a multitude of decision trees during the training phase. Each tree is trained on a bootstrapped sample of the dataset, and at each node, it selects a random subset of features for splitting. The final prediction is made by aggregating the results of individual trees, often through voting.

Random Forest excels at handling high-dimensional data, making it suitable for the DNS dataset with numerous features. It is also robust to overfitting, as it averages out the idiosyncrasies of individual decision trees. Moreover, it provides feature importance scores, allowing for a deeper understanding of the relevance of each attribute in the classification task.

3.7.2 K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) is a straightforward yet effective classification algorithm. It operates on the principle of similarity, where data points are classified based on the class of their nearest neighbors. In other words, if a majority of a data point's k-nearest neighbors belong to a specific class, the data point is assigned to that class.

KNN's simplicity is its strength, making it a suitable choice for identifying malicious DNS requests. It works well when the decision boundaries between classes are not linear or well-defined. KNN is non-parametric, meaning it does not make assumptions about the underlying data distribution, making it versatile for various scenarios.

In the context of DNS request classification, KNN calculates distances between data points, often using Euclidean distance. The choice of the value of k, representing the number of neighbors to consider, is crucial. A small k may lead to noisy results, while a large k may smooth out decision boundaries. Finding the optimal k is part of the model tuning process.

3.7.3 Gradient Boosting:

Gradient Boosting is an ensemble learning method that combines multiple weak learners, typically decision trees, to create a robust classifier. It is known for its high predictive accuracy and adaptability to various data types and distributions.

Gradient Boosting works by iteratively training decision trees, with each subsequent tree aiming to correct the errors of the previous ones. This process continues until the model converges to a strong predictive performance.

In DNS request classification, Gradient Boosting's ability to handle complex relationships between features and the target variable is invaluable. It excels in capturing nuances within the data, making it an excellent choice when precise identification of malicious requests is essential.

In conclusion, each of the classification algorithms mentioned has its unique strengths and characteristics, making them suitable for identifying malicious DNS requests from server log analysis. The choice of algorithm depends on the specific characteristics of the dataset and the desired trade-offs between interpretability, accuracy, and computational complexity.

3.7.4 Logistic Regression:

Logistic Regression is a fundamental classification algorithm that models the probability of an instance belonging to a particular class. It is especially useful when dealing with binary classification problems, such as identifying malicious DNS requests.

Logistic Regression works by applying the logistic (sigmoid) function to a linear combination of input features. This transformation maps the output to the range [0, 1], interpreted as the probability of belonging to the positive class. By setting an appropriate threshold (usually 0.5), instances are classified into one of the two classes.

Logistic Regression's simplicity and interpretability make it a valuable choice for DNS request classification. It provides insight into the relationship between input features and the log-odds of the outcome. Moreover, it can handle high-dimensional data effectively, which is essential for the DNS dataset with numerous features.

3.7.5 Decision Tree:

Decision Trees are intuitive and interpretable classification algorithms that recursively partition the dataset into subsets based on feature conditions. In the context of identifying malicious DNS requests, Decision Trees are valuable for their transparency and ease of interpretation.

A Decision Tree begins with the entire dataset and selects the most informative feature to split the data. This process continues recursively until a stopping criterion is met, such as a maximum depth or minimum samples per leaf. Each leaf node represents a class label.

Decision Trees are well-suited for DNS request classification because they can handle both numerical and categorical data, which is prevalent in this dataset. Their hierarchical structure allows for the visualization of the decision-making process, aiding in the understanding of why certain DNS requests are classified as malicious or benign.

3.7.6 Naive Bayes:

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem, with the "naive" assumption of feature independence. While this assumption may not hold in all cases, Naive Bayes remains a useful tool for DNS request classification.

Naive Bayes works by calculating the probability of an instance belonging to a class based on the joint probabilities of its features. It selects the class with the highest probability as the prediction. Despite its simplicity and the naive independence assumption, Naive Bayes often performs surprisingly well, especially with high-dimensional data.

In the context of DNS request classification, Naive Bayes can efficiently handle the numerous features characterizing each request. It is particularly useful when computational resources are limited, as it requires minimal model training and prediction times.

3.7.7 Support Vector Machine (SVM):

The Support Vector Machine (SVM) is a powerful classification algorithm used extensively in various domains, including cybersecurity. SVM operates on the principle of finding an optimal hyperplane that best separates data points belonging to different classes. In the context of identifying malicious DNS requests from server log analysis with the current data set, SVM proves least effective.

SVM works by mapping data points into a higher-dimensional space, where it strives to create a hyperplane that maximizes the margin between different classes. This margin represents the distance between the hyperplane and the nearest data points of each class, ensuring robust separation. The choice of kernel function is crucial in SVM, as it determines the transformation of data into this higher-dimensional space. Common kernels include linear, polynomial, and radial basis function (RBF).

SVM is particularly suited for this task due to its ability to handle high-dimensional data, making it ideal for the feature-rich DNS request dataset. Its strength lies in its ability to find complex decision boundaries that other algorithms may struggle with. SVM's flexibility in choosing the appropriate kernel function allows for fine-tuning to achieve optimal classification results.

3.7.8 Auto ML (Rapid Miner):

In addition to the manual modeling with various classification algorithms, an AutoML (Automated Machine Learning) tool called RapidMiner was employed in this research. RapidMiner is a robust and user-friendly platform designed to automate the end-to-end process of machine learning, including data preprocessing, model selection, hyperparameter tuning, and evaluation.

RapidMiner operates by systematically testing a range of machine learning algorithms and configurations to identify the most suitable model for the given dataset and problem. This tool streamlines the modeling process by minimizing the need for manual intervention, making it particularly advantageous for handling complex datasets with numerous features. Furthermore, RapidMiner provides insightful visualizations and performance metrics, facilitating the comparison of different models and their suitability for the classification task.

The utilization of RapidMiner in this study allowed for a comprehensive exploration of various classification algorithms and configurations, saving time and resources while ensuring that the most effective models were identified. This approach not only contributes to the accuracy and efficiency of the research but also aligns with the principles of data-driven decision-making in the field of machine learning.

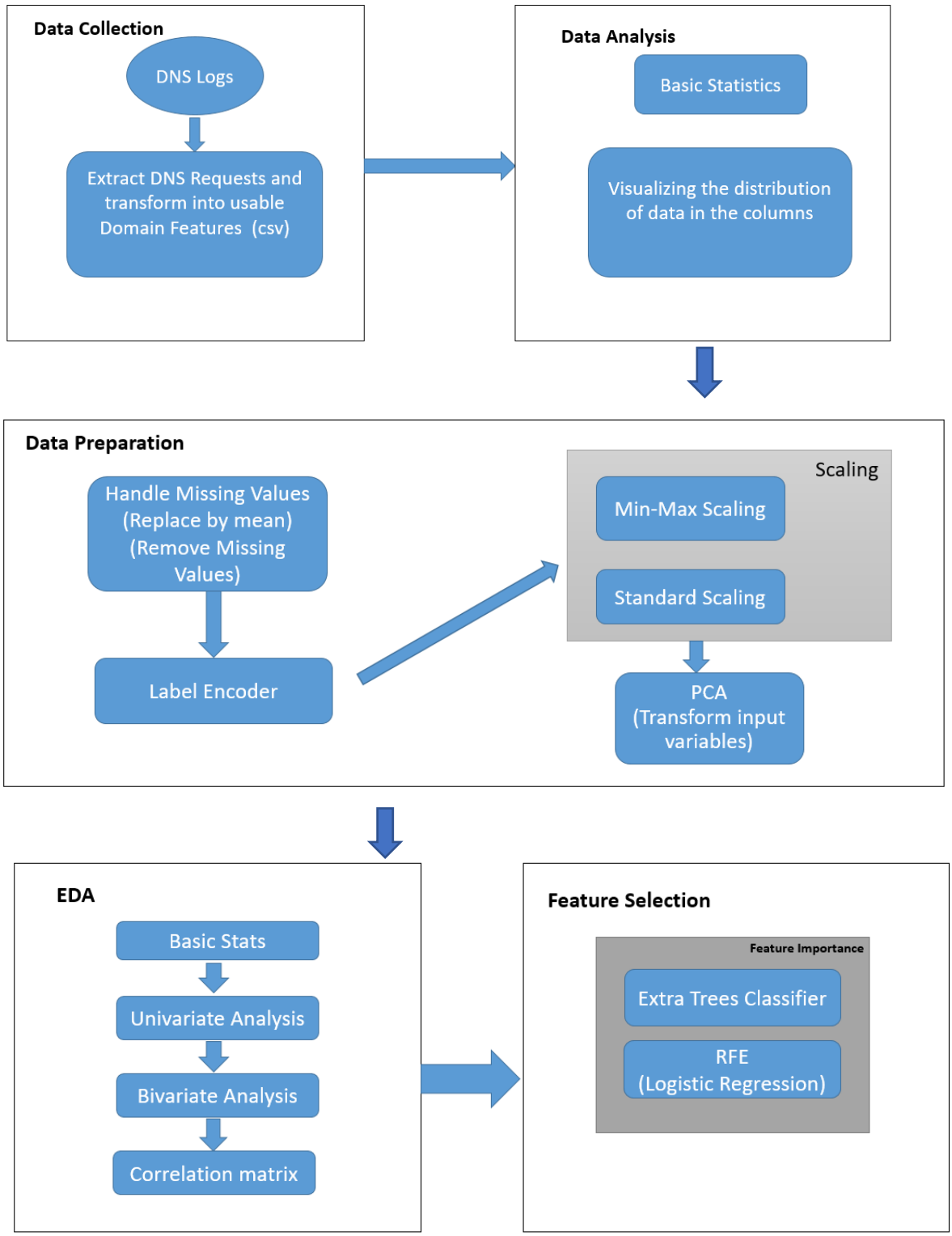
Model	Accuracy	Standard Deviation	Gains	Total Time	Training Time (1,000 Rows)	Scoring Time (1,000 Rows)
Fast Large Margin	0.85333	4.10E-04	18046	1322332	53.59134508	220.9457952
Logistic Regression	0.85321	0.001352174	18030	502211	9.346419808	148.7842962
Random Forest	0.85216	0.001053529	12540	661003	74.4	1397.6
Decision Tree	0.85110	0.001237716	17932	513016	6.93731876	393.7374526
Generalized Linear Model	0.85091	6.47E-04	17922	407805	7.249609636	131.3294669
Gradient Boosted Trees	0.85059	0.001140533	12484	649887	19.168	173.6
Naive Bayes	0.85024	9.05E-04	17888	484400	5.453937096	264.4713362
Deep Learning	0.55507	7.40E-04	2772	627986	274.615213	202.7102387

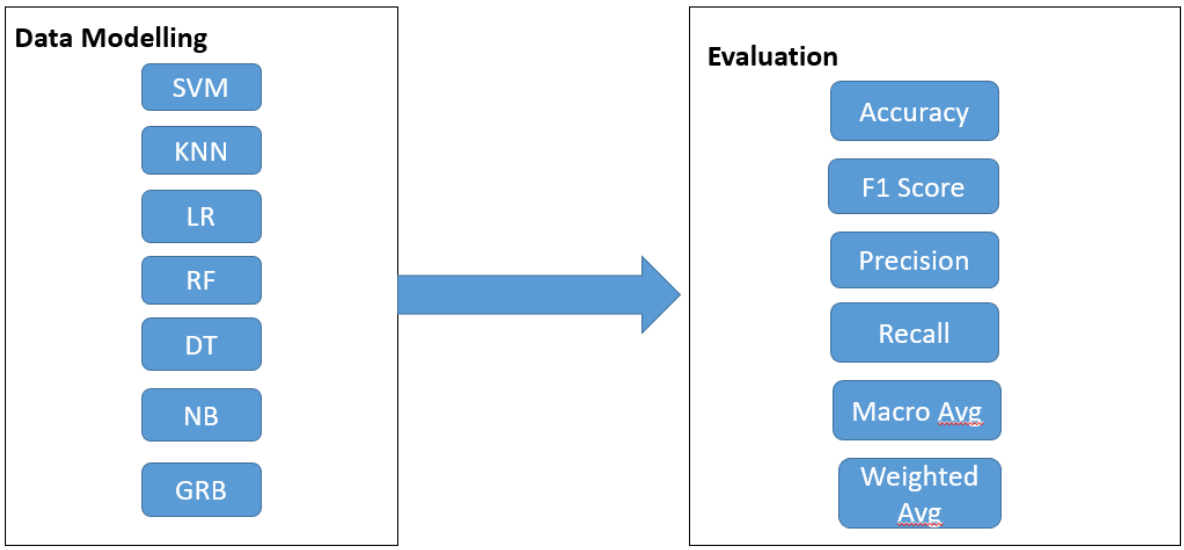
Figure 7: Results from Auto modelling using Rapid Miner

It is notable that most algorithms delivered an accuracy rate hovering around 85%, indicating their consistent ability to effectively distinguish between malicious and non-malicious DNS requests. This uniformity in performance underscores the robustness of the models, irrespective of whether they were manually configured or automatically generated using RapidMiner.

These findings reinforce the reliability of the selected classification algorithms and highlight the potential for leveraging automated modeling tools to streamline and expedite the model development process while maintaining high levels of accuracy.

3.7.9 Data Mining Pipeline




Data Modelling
SVM
KNN
LR
RF
DT
NB
GRB
Evaluation
Accuracy
F1 Score
Precision
Recall
Macro Avg
Weighted Avg

Chapter 4: Evaluation and Results

4.1 Model Evaluation Metrics

In this section, we delve into the various metrics employed to assess the performance of our classification models. The objective was to identify the most effective model for distinguishing between malicious and non-malicious DNS requests. The following evaluation metrics were utilized:

Accuracy: This metric measures the overall correctness of the model's predictions, providing an indication of its reliability.

Precision: Precision quantifies the proportion of true positive predictions out of all positive predictions made by the model. In the context of our research, it reflects how accurately the model identifies malicious DNS requests.

Recall (Sensitivity): Recall assesses the model's ability to correctly identify all actual positive instances (i.e., malicious DNS requests). It helps gauge the model's sensitivity to detecting malicious activities.

F1-Score: The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance, considering both false positives and false negatives.

Area Under the Receiver Operating Characteristic Curve (AUC-ROC): The ROC curve illustrates the model's trade-off between true positive rate and false positive rate at various thresholds. AUC-ROC quantifies the model's capability to distinguish between classes.

Below are the outcomes from the classification reports produced by various classification algorithms employed in this research.

4.2 Classification reports:

4.2.1 Random Forest:

Random Forest demonstrated the highest level of accuracy at 0.89. The corresponding classification report for Random Forest is as follows:

Random Forest Accuracy: 0.890958435571418

Random Forest Report:

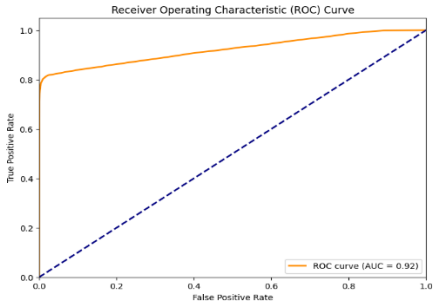
	precision	Recall	f1-score	support
0	0.84	0.96	0.90	13423
1	0.95	0.82	0.88	13475
accuracy			0.89	26898
macro avg	0.90	0.89	0.89	26898
weighted avg	0.90	0.89	0.89	26898

Overall accuracy achieved by Random Forest: 0.89, based on a total support of 26898 instances.

The macro average for precision, recall, and F1-score is 0.90, 0.89, and 0.89, respectively, for the entire dataset, with a total support of 26898 instances.

The weighted average for precision, recall, and F1-score is 0.90, 0.89, and 0.89, respectively, also based on a total support of 26898 instances."

Figure 8: ROC Curve for Random Forest



4.2.2 KNN:

KNN achieved an accuracy of 0.8745, demonstrating its ability to make accurate classifications. In the classification report, KNN exhibited balanced precision, recall, and F1-scores for both malicious (Class 1) and non-malicious (Class 0) categories. This balanced performance suggests that KNN is effective in identifying both types of DNS requests.

KNN Accuracy: 0.8744888095769202

KNN Report:

	precision	Recall	f1-score	support
0	0.83	0.93	0.88	13423
1	0.93	0.82	0.87	13475
accuracy			0.87	26898
macro avg	0.88	0.87	0.87	26898
weighted avg	0.88	0.87	0.87	26898

Gradient boosting: Gradient Boosting achieved an accuracy of 0.8663, showcasing its proficiency in classification tasks. The classification report for Gradient Boosting revealed balanced precision, recall, and F1-scores for both classes.

Gradient boosting Accuracy: 0.8663469402929586

Gradient boosting Report:

	precision	Recall	f1-score	support
0	0.84	0.91	0.87	13423
1	0.90	0.82	0.86	13475
accuracy			0.87	26898
macro avg	0.87	0.87	0.87	26898
weighted avg	0.87	0.87	0.87	26898

4.2.3 Logistic Regression

Logistic Regression attained an accuracy of 0.8578, signifying its competence in classification tasks. The classification report for Logistic Regression exhibited well-balanced precision, recall, and F1-scores for both malicious and non-malicious categories.

Logistic Regression Accuracy: 0.857833296155848

Logistic Regression Report:

	precision	Recall	f1-score	support
0	0.83	0.90	0.86	13423
1	0.89	0.81	0.85	13475
accuracy			0.86	26898
macro avg	0.86	0.86	0.86	26898
weighted avg	0.86	0.86	0.86	26898

4.2.4 Decision Tree

The Decision Tree model achieved an accuracy of 0.8484, indicating its effectiveness in classification tasks. The classification report showed balanced precision, recall, and F1-scores for both malicious and non-malicious DNS requests

Decision Tree Accuracy: 0.8483530374005502

Decision Tree Report:

	precision	Recall	f1-score	support
0	0.86	0.84	0.85	13423

1	0.84	0.86	0.85	13475
accuracy			0.86	26898
macro avg	0.85	0.85	0.85	26898
weighted avg	0.85	0.85	0.85	26898

4.2.5 Naive Bayes

Naive Bayes demonstrated an accuracy of 0.8443, highlighting its proficiency in classification. In the classification report, Naive Bayes exhibited balanced precision, recall, and F1-scores for both malicious and non-malicious categories. This equilibrium in performance suggests that Naive Bayes is well-suited for identifying both types of DNS requests.

Naive Bayes Accuracy: 0.8443378689865417

Naive Bayes Report:

	precision	Recall	f1-score	support
0	0.82	0.88	0.85	13423
1	0.87	0.81	0.84	13475
accuracy			0.84	26898
macro avg	0.85	0.85	0.84	26898
weighted avg	0.85	0.85	0.84	26898

4.2.6 SVM

The Support Vector Machine (SVM) classification model achieved an accuracy of 0.7157 in distinguishing between malicious and non-malicious DNS requests. SVM has demonstrated the least level of accuracy among the classification algorithms

SVM Accuracy: 0.7157037697970109

SVM Report:

	precision	Recall	f1-score	support
0	0.72	0.70	0.71	13423
1	0.71	0.73	0.72	13475
accuracy			0.72	26898
macro avg	0.72	0.72	0.72	26898
weighted avg	0.72	0.72	0.72	26898

Chapter 5: Conclusion

In conclusion, this research has delved into the critical domain of identifying malicious DNS requests through server log analysis using machine learning techniques. The findings and insights obtained from this study shed light on the effectiveness of various classification algorithms in enhancing cybersecurity measures. Key takeaways from this research include:

- **Algorithm Performance:** The evaluation of different classification algorithms revealed that Random Forest exhibited the highest accuracy of 0.89, indicating its capability to effectively differentiate between malicious and non-malicious DNS requests. The choice of algorithm may depend on specific needs, considering the balance between precision and recall.
- **Feature Engineering:** Feature engineering played a pivotal role in model performance. The identification and selection of relevant features were instrumental in the accuracy of the classification models. Feature importance analysis provided valuable insights into the contribution of each feature to the predictive power of the models.
- **Future Enhancements:** The research suggests several avenues for future work. Firstly, the integration of real-time DNS log analysis and continuous model retraining can enhance the system's ability to adapt to evolving threats. Additionally, exploring ensemble methods and deep learning approaches may further improve the accuracy of malicious DNS request detection.

Chapter 6: Future Work

Building upon the foundation laid by this research, future work in the field of identifying malicious DNS requests offers exciting opportunities:

- **Real-time Detection:** Developing systems capable of real-time DNS log analysis and immediate threat response is a critical step towards proactive cybersecurity. Investigating streaming data processing and online learning algorithms can aid in achieving this goal.
- **Ensemble Methods:** Exploring ensemble methods that combine the strengths of multiple classification algorithms can enhance overall model performance. Techniques like stacking and boosting may provide improvements in precision and recall.
- **Deep Learning:** Investigating deep learning architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for DNS request analysis can leverage the power of neural networks to capture complex patterns and behaviors in network traffic.
- **Anomaly Detection:** Extending the research to anomaly detection in DNS traffic can help identify novel and previously unseen threats. Unsupervised learning methods and anomaly detection algorithms can be explored for this purpose.
- **Large-Scale Deployment:** Scaling up the models and systems to handle the demands of large-scale networks and cloud environments is essential. Deployment considerations, such as load balancing and distributed computing, need to be addressed.

References

Du, M. a. L. F. a. Z. G. a. S. V., 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security,* p. 1285–1298.

Jin, Y. a. T. M. a. M. S., 2019. *A Detection Method Against DNS Cache Poisoning Attacks Using Machine Learning Techniques: Work in Progress.* s.l., 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA).

Juvonen, A. a. S. T., 2012. Adaptive framework for network traffic classification using dimensionality reduction and clustering. *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems,* pp. 274-279.

Magalhães, C. M. a. S. M. a. J. P., 2021. DNS dataset for malicious domains detection. *Data in Brief,* 38(2352-3409), p. 107342.

Ming Li, Q. L. G. X. D. G., 2021. Identifying compromised hosts under APT using DNS request sequences. *Journal of Parallel and Distributed Computing,* 152(0743-7315), pp. 67-78.

Pham, T. S. a. H. T. H. a. V. C. V., 2016. Machine learning techniques for web intrusion detection — A comparison. *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE),* pp. 291-297.

Plohmann, D. a. Y. K. a. K. M. a. B. J. a. G.-P. E., 2016. *A comprehensive measurement study of domain generating malware.* s.l.:25th USENIX Security Symposium (USENIX Security 16).

Rajendran, B. a. S. P. a. o., 2018. *Proceedings of the International Conference on Security and Management (SAM).* s.l., The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Rose, R. A. a. R. A. a. M. L. a. D. M. a. S., 2005. *DNS Security Introduction and Requirements, RFC 4033.* s.l.:Internet Engineering Task Force (IETF).

Shalaginov, A. a. F. K. a. H. X., 2016. Malware beaconing detection by mining large-scale dns logs for targeted attack identification. *International Journal of Computer and Systems Engineering,* Volume 10, pp. 743--755.

Shearer, C., Fall 2000. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing,* 5(4), p. 10.

Yan, G. a. L. Q. a. G. D. a. M. X., 2020. Discovering Suspicious APT Behaviors by Analyzing DNS Activities. *Sensors,* Volume 20, p. 731.

Zhang, V.-H. L. a. H., 2022. Log-based anomaly detection with deep learning: How Far Are We?. *Proceedings of the 44th International Conference on Software Engineering,* may.

Appendix

(Data analyzed from DNS logs containing host-based, web-based and numerical features)
34 Features
90K records (two classes)

Table 1: Dataset features with description, data types and default value

			Data Type					
	Feature	**Description**	**Text**	**Boolean**	**Integer**	**Decimal**	**Enumerate**	**Default Value**
1	**Domain**	Baseline DNS used to enrich data (derive features)	X					N/A
2	**DNSRecordType**	DNS record type queried	X					N/A
3	**MXDnsResponse**	The response from a DNS request for the record type MX		X				FALSE
4	**TXTDnsResponse**	The response from a DNS request for the record type TXT		X				FALSE
5	**HasSPFInfo**	If the DNS response has Sender Policy Framework attribute		X				FALSE
6	**HasDkimInfo**	If the DNS response has Domain Keys Identified Email attribute		X				FALSE
7	**HasDmarcInfo**	If the DNS response has Domain-Based Message Authentication		X				FALSE
8	**IP**	The IP for the domain	X					null
9	**DomainInAlexaDB**	If the domain it's registered in the Alexa DB		X				FALSE

10	**CommonPorts**	If the domain it's available for common ports (80, 443, 21, 22, 23, 25, 53, 110, 143, 161, 445, 465, 587, 993, 995, 3306, 3389, 7547, 8080, 8888)		X				FALSE
11	**CountryCode**	The country code associated with the IP of the domain	X					null
12	**RegisteredCountryCode**	The country code defined in the domain registration process (WHOIS)	X					null
13	**CreationDate**	The creation date of the domain (WHOIS)					X	0
14	**LastUpdateDate**	The last update date of the domain (WHOIS)					X	0
15	**ASN**	The Autonomous System Number for the domain			X			-1
16	**HttpResponseCode**	The HTTP/HTTPS response code for the domain					X	0
17	**RegisteredOrg**	The organization name associated with the domain (WHOIS)	X					null
18	**SubdomainNumber**	The number of sub-domains for the domain			X			0
19	**Entropy**	The Shannon Entropy of the domain name			X			0
20	**EntropyOfSubDomains**	The mean value of the entropy for the sub-domains			X			0
21	**StrangeCharacters**	The number of characters different from [a-zA-Z] and considering the existence maximum of two			X			0

		numeric integer values						
22	**TLD**	The Top Level Domain for the domain	X					null
23	**IpReputation**	The result of the blocklisted search for the IP		X				FALSE
24	**DomainReputation**	The result of the blocklisted search for the domain		X				FALSE
25	**ConsoantRatio**	The ratio of consonant characters in the domain				X		0
26	**NumericRatio**	The ratio of numeric characters in the domain				X		0
27	**SpecialCharRatio**	The ratio of special characters in the domain				X		0
28	**VowelRatio**	The ratio of vowel characters in the domain				X		0
29	**ConsoantSequence**	The maximum number of consecutive consonants in the domain			X			0
30	**VowelSequence**	The maximum number of consecutive vowels in the domain			X			0
31	**NumericSequence**	The maximum number of consecutive numerics in the domain			X			0
32	**SpecialCharSequence**	The maximum number of consecutive special characters in the domain			X			0

33	**DomainLength**	The length of the domain			X			N/A
34	**Class**	The class of the domain (malicious = 0 and non-malicious = 1)			X			N/A

Table 2: Values description for enumeration features where X denotes all possible values

S.no	Feature	Values description
1	CreationDate	Without data = 0
		Until one month = 1
		Until six months = 2
2	LastUpdateDate	Until one year = 3
		After one year = 4
3	HttpResponseCode	Without data = 0
		1XX response = 1
		2XX response = 2
		3XX response = 3
		4XX response = 4
		5XX response = 5