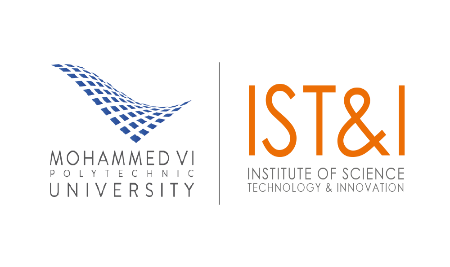
****  

***REPORT OF THE 1st SCIENTIFIC INTERNSHIP FOR THE TRAINING ON SCIENTIFIC AND INDUSTRIAL RESEARCH***

***(From 03 January to 31 May 2022)***

***THEMATIC: Preventing phishing attacks Using A.I.* **

# *TITLE OF THE INTERNSHIP SUBJECT*

# ***Network Packet level –based Intelligent Phishing Intrusion Detection System (NIPDS)***

***INSTITUTE OF SCIENCE, TECHNOLOGY AND INNOVATION (IST&I)***

***UM6P-CS: School of computer Science***

***Protractors of my internship work: Prof. Ahmed RATNANI and Prof. Achraf EL ALLALI Academic year: 2021-2022***

|  |  |
| --- | --- |
| Realized by | Supervised by |
|  |  |

***Khaoula Hidawi******Dr. Ismail Berrada***

***Acknowledgments***

*I would like to thank my esteemed supervisor Dr. Ismail Berrada for his invaluable supervision, support, and tutelage during the course of this scientific internship. My gratitude extends to The OCP Group (Office Chérifien des Phosphates) for the funding opportunity to undertake my internship at the Department of Computer Science within Mohammed VI Polytechnic University. Additionally, I would like to express gratitude to Mr. Berrada for his treasured support which was really influential in shaping my experiment methods and critiquing my results and for his mentorship. I would like to thank my internship mates, colleagues, and research team for a cherished time spent together in the meetings, and in social settings. My appreciation also goes out to the Institute Of science, technology and innovation for this funded predoctoral fellowship opportunity; Madame Fouad, my family and friends for their encouragement and support all through my studies*.

***Contents***

[***Network Packet level –based Intelligent Phishing Intrusion Detection System (NIPDS)*** 1](file:///F:\UM6P\Report\RAPPORT%20d'état%20d'avancemenr%20DE%20STAGE%20%20FRSI_Khaoula_Hidawi.docx#_Toc104904576)

[*A.* *Introduction* 5](#_Toc104904577)

[***I.*** ***The Research Problem*** 5](#_Toc104904578)

[**1.** **Keywords:** 9](#_Toc104904579)

[***II.*** ***Planned Research Methodology*** 9](#_Toc104904581)

[*B.* *State of the Art: Review of Related Literature* 10](#_Toc104904582)

[*C.* *Preparing of Research Design* 22](#_Toc104904583)

[**1.** **Features Extraction:** 25](#_Toc104904584)

[1.1 Lexical features 25](#_Toc104904585)

[1.2 Third party features 31](#_Toc104904586)

[*D.* *Data Collection* 36](#_Toc104904587)

[1. Train Dataset: CIC-Bell-DNS 2021 36](#_Toc104904588)

[2. Test Dataset: real time based scenario 38](#_Toc104904589)

[*E.* *Models Implementation* 45](#_Toc104904590)

[*F.* *Preliminary Results and Analysis* 54](#_Toc104904591)

[G. *Conclusion and* Future Work 68](#_Toc104904596)

[H. Bibliographic references 71](#_Toc104904597)

***List of Figures***

[Figure 1- Most Target Industries in Phishing Cyber-attacks 5](#_Toc104904812)

[Figure 2- HTTPS Phishing by Year 6](#_Toc104904813)

[Figure 3- Data shown in HTTP requested Header 7](#_Toc104904814)

[**Figure 4- Data shown in HTTPS header.** The HTTP Urls shows the full path but in real data flow encrypted HTTPS request protects most things: This is the same for all HTTP methods (GET, POST, PUT, etc.). The URL path and query string parameters are encrypted, as are POST bodies. **We can only view domain names.** 7](#_Toc104904815)

[Figure 5- Phishing Attack Graph 9](#_Toc104904816)

[Figure 6- Classical Phishing Models 11](#_Toc104904817)

[Figure 7 - Summary of currently used anti-phishing systems 21](#_Toc104904818)

[Figure 8 - Design sketch for INPDS 22](#_Toc104904819)

[Figure 9- Taxonomy of an IDS System 23](#_Toc104904820)

[Figure 10 - Packet-Based NIDS 24](#_Toc104904821)

[Figure 11 - Flow-Based NIDS 25](#_Toc104904822)

[Figure 12 - FEATURE AVERAGE MERIT 31](#_Toc104904823)

[Figure 13 - Statistics of the domains dataset 37](#_Toc104904824)

[Figure 14- Processed Training Dataset CSV file contains 5012 586 row of benign and phishing Domains and their features 38](#_Toc104904825)

[Figure 15- Design for generating synthetic test datasets that represent a real time packet flow scenario 39](#_Toc104904826)

[Figure 16- URLibRequest.py source code to open connexion to the List of URLs 40](#_Toc104904827)

[Figure 17- Outputs of the URLibrequet.py in kali lunix terminal 41](#_Toc104904828)

[Figure 18- The outputs of Wireshark and passer.py in terminal 2 41](#_Toc104904829)

[Figure 19- Wireshark Packet capture outputs of the 10 000 url connexions 42](#_Toc104904830)

[Figure 20- terminals outputs for passer.py and urlibrequest.py 42](#_Toc104904831)

[Figure 21 - Resulted Testing Dataset with 2126 entries for benign and phishing Domain names and its 22 DNS-based extracted features 44](#_Toc104904832)

[Figure 22 - Soft and Hard conversions of categorical features used in our models next to One-hot encoder 61](#_Toc104904833)

# *Introduction*

## ***The Research Problem***

The Internet has become an indispensable component of our daily social and financial lives. Nonetheless, internet users may be subject to a variety of web threats that can result in financial losses, identity theft, data loss, and brand reputation damage be it the public or private sectors. Phishing is a type of web threat and cybercrime defined as the act of imitating a legitimate company's website in order to steal sensitive information such as usernames, passwords, and social security numbers.

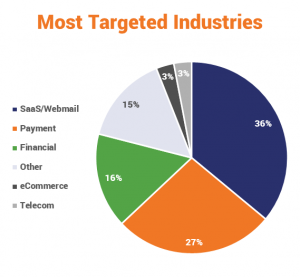


Figure 1- Most Target Industries in Phishing Cyber-attacks [5]

So far, there is no single solution that can capture every phishing attack at a network-based level for real HTTPS data flow, According to a report **[1]** by the Anti-Phishing Working Group (APWG) and contributor Phish Labs, in the first quarter of 2021, 83% of phishing sites had SSL encryption enabled. In this study, we introduced a unique intelligent models for predicting phishing attacks. Network-based intrusion detection systems that monitor cutting-edge high-volume network linkages need greater computing resources than ordinary computer hardware can provide. Possible use cases for merging traditional, packet-based, and innovative, flow-based intrusion detection are described using this model. An increasing amount of web traffic is currently encrypted using HTTPS. While most of the HTTPS traffic is legitimate, a growing slice is generated by malware. The use of the HTTPS protocol by malware and phishing attacks makes its detection more challenging.

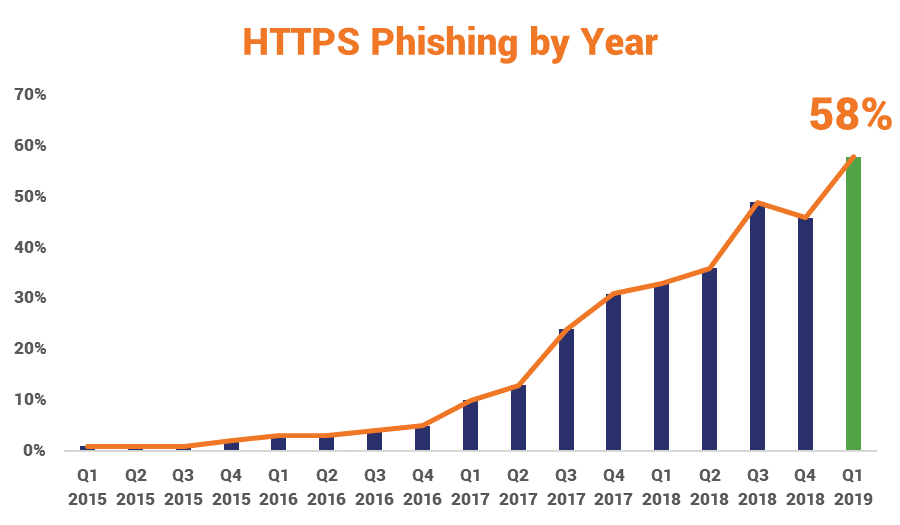


Figure 2- HTTPS Phishing by Year [5]

The current strategy in the literature is to use HTTPS interceptor proxies to identify HTTPS malware traffic **[4]**. This technology necessitates on-the-fly decryption of traffic, which poses certain risks to data and communication security and privacy.

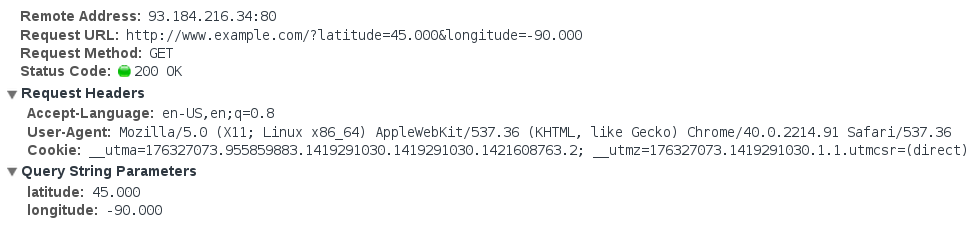
******

Figure 3- Data shown in HTTP requested Header

******

**Figure 4- Data shown in HTTPS header.** The HTTP Urls shows the full path but in real data flow encrypted HTTPS request protects most things: This is the same for all HTTP methods (GET, POST, PUT, etc.). The URL path and query string parameters are encrypted, as are POST bodies. **We can only view domain names.**

The purpose by the end of this research project is to detect HTTPS/HTTP malicious phishing traffic without decryption by analyzing real-time device traffic and large network captures in the form of a PCAP file to extract network traffic characteristics. We propose a novel detection model that makes use of the underlying DNS traffic features that is put into a machine learning classifier. A solid feature engineering mechanism plays a pivotal role in boosting the performance of any machine learning model. Therefore, we have extracted effective and practical features from DNS traffic categorizing them into groups of lexical-based and third-party based features. Third-party features are biographical information about a specific domain extracted from third-party APIs. **[2]** **[3]**. Experimental evaluation is conducted using a CSV public dataset that we preprocessed and generated while the model training data is taken from CIC-Bell-DNS 2021 Dataset, which is a collaborative project with Bell Canada (BC) and Cyber Threat Intelligence (CTI) **[18]**. In their work, they generated and released a large DNS features dataset of 400,000 benign and 13,011 malicious samples processed from a million benign and 51,453 known-malicious domains from publicly available datasets. The malicious samples span between three categories of spam, phishing, and malware - For our research project I only worked with the phishing samples. The CIC-Bell-DNS2021 was the best decision for this research because it replicates the real-world scenarios with frequent benign traffic and malicious domain types **[18]**.

### **Keywords:**

### Web threats, HTTPS Phishing attack, HTTP/HTTPS protocol, Information security, neural network, Data mining, Network-based IDS, encrypted Network traffic, Cyber Threat intelligence, Machine learning ,DNS traffic.

## ***Planned Research Methodology***

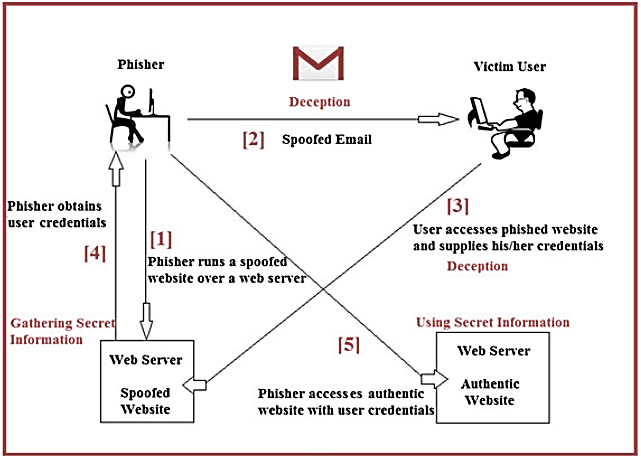


Figure 5- Phishing Attack Graph

The study methodology outlines the steps we toke to try to achieve our goal of implementing a Packet-Based Intelligent Network phishing Intrusion Detection system (INPDS).

The plan includes the processes and methods we followed, as well as materials relating to the topic to be investigated.

***The research problem*** - The first and most important stage in the research process is to define the research topic.

**Review of Related Literature** – This is necessary to familiarize oneself with existing research materials linked to the research problem which is identification of phishing attacks in Networks packet-based IDS. The review for the most important intelligent models in phishing classification can be view in Section B (this step took approximately one month period).

**Preparing of research design** – The research design is the theoretical framework within which the research will be carried out. It serves as a guide for the gathering of data, in our instance of implementing a Packet-Based Intelligent Network phishing Intrusion Detection system. The idea of the design is to use machine learning to classify Network packets to benign and phishing in real-time flow (for both http/https protocol) based on DNS records and domain name features. It operates by using a pre-programmed list of known phishing threat features and their indicators of compromise (IOCs). As a signature based INPDS it will monitor the packets traversing the network, it compares these packets to the database of known IOCs or attack signatures to flag any suspicious behavior. (This step took approximately a month and a half).

**Collection of data** – After formulating the research problem, developing a research design and identifying the methods and procedures to be used, the next step was collecting sufficient data from which we can draw conclusions and inferences for the study (This step took a month to find the right training set and generating the test datasets).

**Analysis of data** – This involves processing and analyzing the collected data then organizing it in a manner where the information can answer the question or give solution to decreasing phishing attacks in real time flow using machine learning models.

**Generalization** – Here, we puts the summary of all the findings of the research. Recommendations was also be included in this section of the research methodology.

**Presentation of results** – The research will then be presented to a panel to defend the relevance of the study conducted by contrast to exciting phishing classification models.

## *State of the Art: Review of Related Literature*

The following figure lists the experimental results of some existing models on detecting phishing attacks in the literature. For the generality of comparisons, four kinds of phishing detection methods were selected in the experiment, including the phishing detection system proposed by Ma (abbreviated as ‘‘Ma’’ in this table) based on the blacklist; the Cantina model based on the heuristic method; the tool proposed by Zhang (abbreviated as ‘‘Zhang’’ in this table) based on the visual similarity method; the Off the-Hook with the results being the averages often repetitive experiments on the two datasets. We can see that, due to strong learning ability and high classification accuracy, the performances of neural network classifiers (ANN and DTOF-ANN) are better than the other classifiers in the literature, in this section I reviewed the best-classified models for phishing detection, their limitations, implementations, and findings:

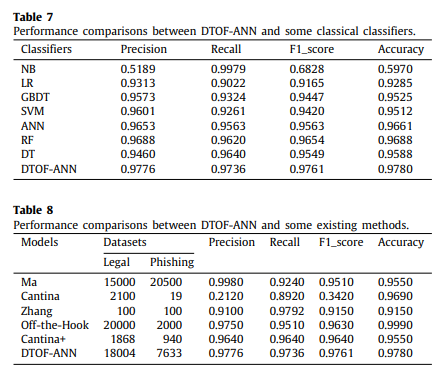


Figure 6- Classical Phishing Models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Research Article | *Problem Addressed / Identified* | Research Contribution | Novelty/Rationale and  Significance: | Limitations and Weaknesses | Implementation  Details | Findings and Conclusions |
| *URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection [6]* | *Malicious URLs host unsolicited content and are used to perpetrate cybercrime traditionally, this is done through the usage of blacklists, which cannot be exhaustive, and cannot detect newly generated malicious URLs****.*** *To address this, recent years have witnessed several efforts to perform Malicious URL Detection using Machine Learning. The most popular and scalable approaches use lexical properties of the URL string by extracting Bag-of-words like features, followed by applying machine learning models such as SVMs. There are also other features designed by experts to improve the prediction performance of the model. These approaches suffer from several limitations: (i) Inability to effectively capture semantic meaning and sequential patterns in URL strings; (ii) Requiring substantial manual feature engineering; and (iii) Inability to handle unseen features and generalize to test data.* | *To address these challenges, the researchers propose URLNet, an end-to-end deep learning framework to learn a nonlinear URL embedding for Malicious URL Detection directly from the URL. Specifically, they apply Convolutional Neural Networks to both characters and words of the URL String to learn the URL embedding in a jointly optimized framework.* | *URLNet allows us to alleviate the shortcomings of traditional approaches such that Character and Word CNNs automatically identify and learn the semantic and sequential patterns in which the characters and words appear in the URL; as well as* *automatically learns features to represent the URL, and we do not rely on any other complex or expert features for the learning task; and also The model learns patterns based on both character and word embedding. Due to the limited number of characters, this character embedding can generalize to new URLs easily. For word-embedding, even if the test URLs contain new unseen words, the character-based (advanced word) embedding of the words still allows us to obtain representation for these new words. This way URLNet has superior generalization ability compared to existing approaches* | *In this research the authors focused primarily on the lexical features to obtain the feature representation for the URLs. The most popular features were Bag of Words, Term Frequency features or n-gram features .However, the authors argue that none of these methods effectively capture the sequential properties of the URL string (or substrings). Moreover these methods fail to extract useful information from unseen words in the test URLs.* | *The model comprises two branches with CNNs.* ***T****he first branch is a Character-level CNN where the character embedding is used to represent the URL. The second is the Word-level CNN where a word-level embedding is used to represent the URL. The word embedding itself is a combination of the individual word’s embedding and the character-level embedding of that word.* ***Dataset Collection.*** *They collected a large corpus of labeled URLs from VirusTotal an antivirus group whose services are often used to validate whether a given query URL is malicious or not.* | *The article proposed Character CNNs and Word CNNs for this task, and jointly optimized the network. + they proposed advanced word-embedding techniques which are particularly useful to deal with rare words, a problem usually observed in malicious URL Detection tasks (and not in traditional NLP tasks)+ This approach also allowed URLNet to learn embedding from unseen words at test time, and exploit sub word information.* |
| *CANTINA+: A Feature-rich Machine Learning Framework for Detecting Phishing Web Sites [7]* | *Typically, phishing detection methods either use human verified URL blacklists or exploit webpage features via machine learning techniques. However, the former is frail in terms of new phish, and the latter suffers from the scarcity of effective features and the high false positive rate (FP).* | *To alleviate those problems, The authors propose a layered anti-phishing solution that aims at 1) exploiting the expressiveness of a rich set of features with machine learning to achieve a high true positive rate (TP) on novel phish, and 2) limiting the FP to a low level via filtering algorithms. Specifically, the article proposed CANTINA+, a comprehensive feature-based approach in the literature including eight novel features, which exploits the HTML Document Object Model (DOM), search engines and third party services with machine learning techniques to detect phishing.* | *An ideal anti-phishing solution needs to have reasonable TP against new attacks with very low FP while involving minimum manual labor.**The key to achieve a high TP is to design new features that are characteristic of phishing patterns, and the core ingredient leading to a very low FP is filtering via heuristics****.*** *With those in mind,* ***+*** *this research goal in this paper is contributing to the literature by addressing the weaknesses of both blacklists and feature-based methods in a unified framework. By proposing novel features to improve the TP and design filtering algorithms absent in the literature to reduce FP and human effort.* | *Cross site scripting (XSS) is a type of typical security vulnerability in web applications by which criminals may launch a variant of phishing attack.**This anti-phishing solution is certainly not a panacea, but it is not designed to deal with this type of attacks in the first place. In this case, the authors suggest that they will rely on other techniques such as cookie protection to assist our approach in detecting phish.* | *In the training stage, 15 feature values are extracted from each instance in the training corpus; the feature values are organized in proper format and forwarded to the ML engine; Then classifiers are built for phish detection. In the testing stage, the hash-based filter examines whether or not the incoming page is a near duplicate of known phish based on comparing SHA1 hashes; if no hash match is found, the login form detector is called, which directly classifies the webpage as legitimate if no login form is identified; the webpage is sent to the feature extractor when a login form is detected;* | *in the randomized evaluation where training and testing phish are randomly selected, CANTINA+ achieved over 92% TP on unique testing phish, over 99% TP on near-duplicate testing phish, and about 0.4% FP under 10% training phish with login form filtering to slash false positives. In the time-based evaluation methodology where earlier phish were used as training data to build classification models for later attacks, the approach achieved over 92% TP on unique testing phish, over 99% TP on near-duplicate testing phish, and about 1.4% FP under 20% training phish with a two-week sliding window.* |
| *Cat BERT: Context-Aware Tiny BERT for Detecting Targeted Social Engineering Email [8]* | Social engineering attacks leveraging hand-crafted emails are a major threat vector today. Because these emails are often hand-written, individually targeted, and incorporate background research on their targets, they pose *a significant challenge for conventional detection systems which reply on spam-like duplication between previously seen and new malicious emails.* | *To address the challenges, the paper propose a phishing detection strategy based on transformers , leveraging a BERT-derived approach that is trained on a self-supervised cloze task on a public corpus of documents and then optimizing the language model to perform phishing detection****.*** *This transfer-learning procedure allows the network to learn a useful representation of natural language syntax and semantics before specialization in phishing detection.* | *Model leverages both natural language and email header inputs, is more computationally efficient than competing transformer approaches, plus it is less prone to adversarial attacks which deliberately replace keywords with typos or synonyms.* | *Distil-BERT has 135 million parameters, which is the largest model with 6 Transformer blocks, and a correspondingly long inference time. Cat-BERT with 3 Transformer blocks has 117 million parameters, is about 15% smaller than Distil-BERT and obtains 1.6x speed up in CPU inference time (using an AWS m5.large instance type). The relatively modest reduction in parameter size for Cat BERT is due to the fact that all three models use a large embedding layer with 92 million parameters, which accounts for about 70% of all parameters.* | *The proposed Cat-BERT model is downsized from Distil-BERT by taking odd-numbered Transformers and replacing missing Transformers with simple Adapters, which we find reduces model size with no cost to accuracy. The adapters consists of two dense and a ReLU activation units with a residual connection. The content input from email text is fed to the Embedding block and the context input from email headers is combined in the classification head in Cat-BERT.* | *They conducted experiments with two non-BERT models, Long Short-Term Memory (LSTM) and Logistic Regression (LR). Also baselined with Distil-BERT, trained with Adam and with a class-balanced batch size of 128. Cat-BERT, proposed model, has three Transformer blocks with content and context features. As seen from figure 2. It outperform all the other models.* |
| *DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features [9]* | *Duplicate points in public datasets and negative and useless features in the feature vectors trap the training of neural networks into the problem of over-fitting, which will make the trained classifier weak when detect phishing websites.* | *This paper proposes DTOF-ANN (Decision Tree and Optimal Features based Artificial Neural Network) to tackle this shortcoming, which is a neural-network phishing detection model based on decision tree and optimal feature selection.* | *The traditional clustering algorithm is improved with an incremental selection of initial centers to remove the duplicate points from the public datasets. Plus, an optimal feature selection algorithm based on the new defined feature evaluation index, decision tree and local search method is designed to prune out the negative and useless features. The optimal structure of the neural network classifier is constructed through properly adjusting parameters and trained by the selected optimal features.* |  | *The overall workflow of DTOF-ANN, which can be divided into two stages: the training stage and the testing stage. The former stage first introduces the improved K-medoids algorithm to delete the duplicate data points in the open phishing datasets. Then, the optimal feature selection algorithm is designed to construct the optimal feature vector for the underlying neural network classifier, where the new index, F value, is defined to evaluate the importance of phishing features of data points in the phishing datasets.* | *The Blacklist module improves the performance, which is constructed by selecting data points from the Alexa website and the Phish Tank website. The tested URLs are firstly checked by querying if they are already existed in the Blacklist. The target URLs are directly marked as the phishing ones if so, and, otherwise, the optimal feature vectors of the corresponding URLs are processed by the trained neural network classifier. The detected phishing URLs are also stored in the Blacklist to avoid the same kinds of URL in the future processed repeatedly.* |
| *Off-the-Hook: An Efficient and Usable Client-Side Phishing Prevention Application [10]* | *The state-of-the-art solutions in the literature have good performance, but they suffer from several drawbacks including potential to compromise user privacy, difficulty of detecting phishing websites whose content change dynamically, and reliance on features that are too dependent on the training data.* | *To address these limitations the authors present a new approach for detecting phishing webpages in real-time as they are visited by a browser. Their approaches relies on modeling inherent phisher limitations stemming from the constraints they face while building a webpage ,Off-the-Hook, exhibits several notable properties including high accuracy, brand-independence and good language-independence, speed of decision, resilience to dynamic phish and resilience to evolution in phishing techniques.* | *Introducing a new phish detection tool, Off-the-Hook, implemented as a browser add-on that can decide in real time if a visited webpage is a phish. On encountering a phish, Off-the-Hook identifies the target brand mimicked by the phish. Off-the-Hook implementation is fully-client-side and the decision process relies solely on information extracted from the web browser while loading a webpage.* | *The authors stated that: “The interpretation and appreciation of the safe toast notification were less clear, as some considered it redundant but others found it helpful. No user correctly guessed why it appears on some sites and not others, which is potentially problematic. Removing the safe toast notification would be a possible remedy for this, given that the same information is also present at the navigation bar. The main usability problem with this would be that the loading screen may result in confusion or suspicion, since it currently has no information.* | *Off-the-Hook is fully implemented on the client-side. It computes its decision based on the webpage content actually displayed in the browser. It is thus privacy preserving and is not vulnerable to dynamic phishes. The features used for the model are non-static: they represent phisher limitations rather than static words found in training data. The detection is thus context-independent* | *Preserves users’ privacy, provides real-time protection and is resilient to dynamic phish since the content actually loaded in the browser is analyzed to render a decision.* |
| *CANTINA: A Content-Based Approach to Detecting Phishing Web [11]* | *Based on the evaluation conducted by the authors of ten anti-phishing tools for previous study, They found that only one tool could consistently detect more than 60% of phishing web sites without a high rate of false positives.* | *The paper present the design, implementation, and evaluation of CANTINA a novel content-based approach for detecting phishing web sites. CANTINA examines the content of a web page to determine whether it is legitimate or not, in contrast to other approaches that look at surface characteristics of a web page, for example the URL and its domain name* | *Authors argue that there is a strong need for better automated detection algorithms that reduce false positives (incorrectly labeling legitimate web sites as phishing), while lowering phish detection rates only slightly.* | *CANTINA suffers from performance problems due to the time lag involved in querying Google. Another issue is the potential for attackers to circumvent CANTINA. Phishing is an arms race, with criminals continually devising new ways of tricking people. So the authors have identified three direct approaches by which CANTINA might be attacked.* | *CANTINA makes use of TF-IDF for detecting phishing sites. TF-DF is a well-known information retrieval algorithm that can be used for comparing and classifying documents, as well as retrieving documents from a large corpus. TF-IDF yields a weight that measures how important a word is to a document in a corpus. The importance increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus* | *The result of the experiments shows that the pure TF-IDF approach can catch about 97% phishing sites with about 6% false positives, and after combining some simple heuristics they were able to catch about 90% of phishing sites with only 1% false positives.* |

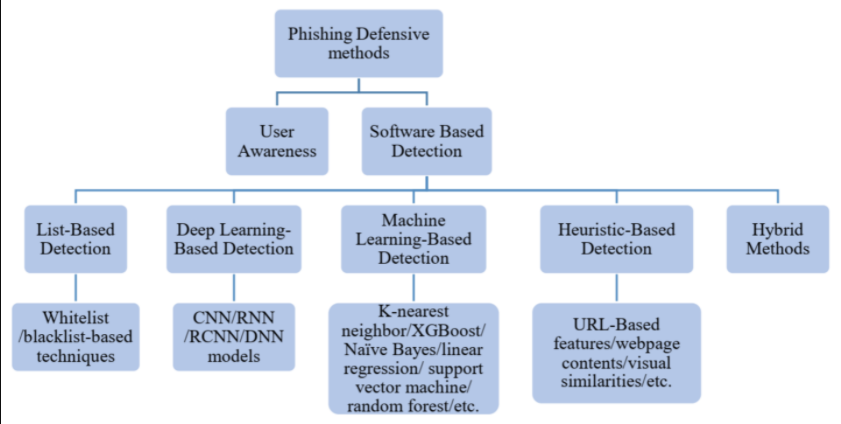


Figure 7 - Summary of currently used anti-phishing systems

# *Preparing of Research Design*

The idea behind the Design is using machine learning to classify Network packets to benign and phishing in real-time flow based on DNS and domain name features. The initial thought is that INPDS will operates by using a pre-programmed list of known phishing threat features and their indicators of compromise (IOCs). As a signature based-IDS it will monitor the packets traversing the network then compares these packets to the database of known IOCs or attack signatures to flag any suspicious behavior.

Designing an efficient and applicable vetting system that identifies malicious domains in real-time is of paramount importance. To this end, we provide a scheme of our proposed deployment model that predicts the label of a domain in real-time through a real-time detection server (figure 7) inspired by the article in [19]. First, the user submits a domain to the web client. The request is forwarded to the real-time server where an HTTP request is sent to the domain, the associated DNS packets are captured, and 22 features (***currently***) are derived from the packets. Then, the trained model predicts the category of the domain and returns the prediction. [19].

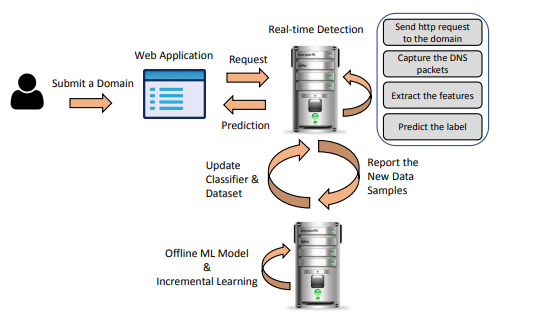


Figure 8 - Design sketch based on [19] for INPDS

Works similar to anti-virus software. It employs a signature (pattern that correspond to a known threat) database of know attacks, and if a successful match with current input, an alert is raised. A well-known example of this type is Snort which is an open source IDS that monitors network by matching each packet it observes against a set of rules of the DNS message in a specific packet window. Then, for each captured domain, we extract the lexical and third party features. As for the second phase of the design, also inspired by article [19] the new arrived data samples will be reported to the offline server which holds a copy of the ML model. The offline server needs to work with data streams and thus it applies incremental learning and hyper parameter optimization. In the scheme, incremental learning realizes by re-training the same model and tunes it according to new arrival data samples. Since incremental learning learns each new data sample individually, the whole process requires low computational cost. In the last step, the classifier and dataset residing on a real-time detection server are updated accordingly to serve coming data streams in the future.

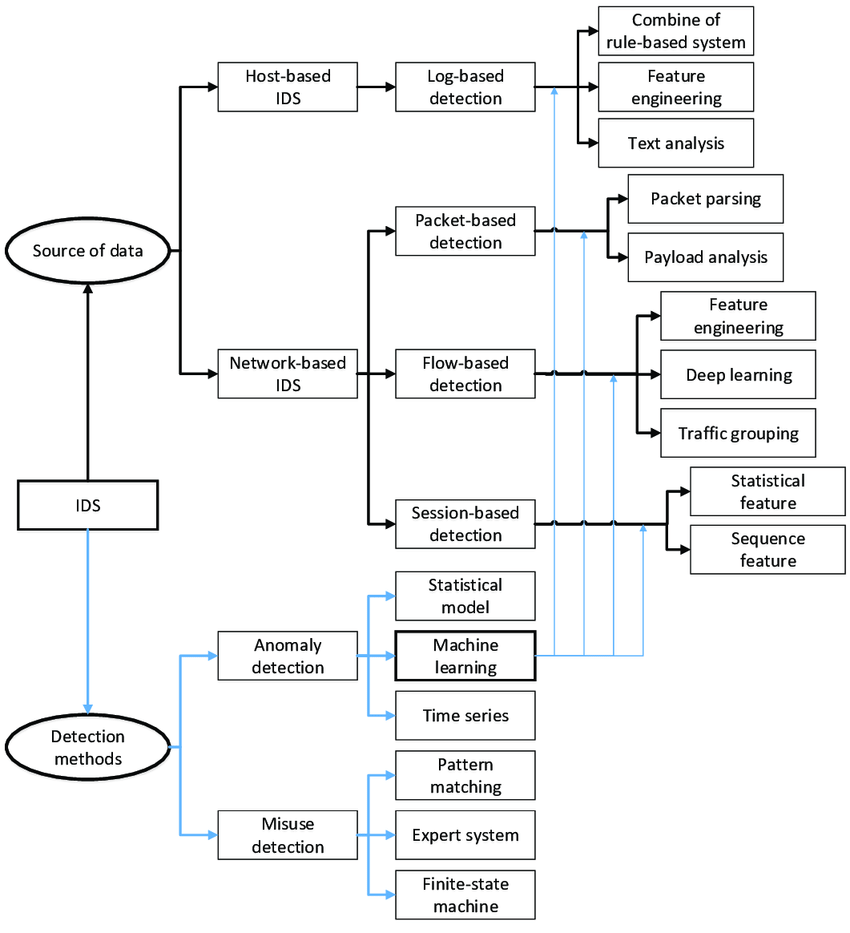
******

Figure 9- Taxonomy of an IDS System

Network Intrusion Detection Systems (NIDSs) are widely-deployed security tools for detecting cyber-attacks and activities conducted by intruders for observing network traffics. With the increase in network speed and number and types of attacks, existing NIDSs, face challenges of capturing every packet to compare them to malicious signatures. These challenges will impact on the efficiency of NIDSs, mainly the performance and accuracy power. Throughout the analysis of the literature on this topic, the article in [20] found that the Packet-based NIDSs process every packet (payload) received. While it produces low false alarms, it is very time consuming, therefore it is hard, to perform packet-based approach at the speed of multiple Gigabits per second (Gbps). Flow-based NIDSs have an overall lower amount of data to be process, therefore it is the logical choice for high speed networks but it suffers from producing high false alarms [20]. Therefore, it can be recommended that, a hybrid or a mixture model of both NIDSs may ensure a higher ability to react on the wider scope of attacks within the high-speed networks environment, using packet-based phishing detection system for HTTP /HTTPs protocol is only my side of the project where my other research colleagues are working on flow-based IDS and other types of attacks.

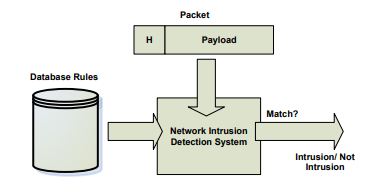


Figure 10 - Packet-Based NIDS

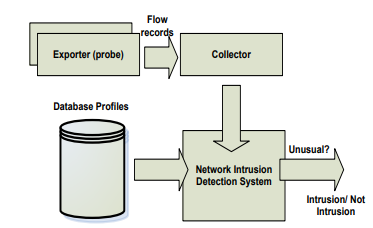


Figure 11 - Flow-Based NIDS

## **Features Extraction:**

The following is an in depth analysis of each feature currently used for our INPDS model:

# Lexical features

Lexical features help detect malicious domain names since attackers apply different typo squatting and obfuscation methods to mimic the real domain names. In this research, we have extracted 11 features from each domain elaborated as follows:

**Subdomain:** Most malicious domains have few subdomains to mimic the real domain. For instance, to mimic www.facebook.com, the malicious domain could be in the format of www.facebook.f.com. In this case, the Top-Level Domain (TLD) is f and not facebook. However, people without enough background knowledge may be deceived.

They give the example of the following link: ***http://www.hud.ac.uk/students/.*** A domain name might include the country-code top-level domains (ccTLD), which in this example is “uk”. The “ac” part is shorthand for “academic”, ***the combined “ac.uk” is called a second-level domain (SLD)*** and **“hud” is the actual name of the domain.**

To produce this rule for extracting this feature:

**First we** had to omit the (www.) from the URL which is in fact a sub domain in itself.

**Then,** they removed the (ccTLD) i.e : country-code and top-level domains if it exists.

**Finally,** they counted the remaining dots in the URL.

***If*** the number of dots is greater than one,

**then** the URL is classified as “**Suspicious**” since it has one sub domain.

***However, if*** the dots are greater than two, it is classified as “***Phishing***” since it will have multiple sub domains.

***Otherwise,*** if the URL has no sub domains, we assign “***Legitimate***” to the feature.

--By the end on the analysis we concluded the following rule:

***Rule***: IF

**Top-Level Domain (TLD):** Based on the previous studies, most malicious domains are with specific TLDs for a long duration of time, such as .com, .pw, to name a few. The possibility of a URL being malicious is higher when its TLD historically has been used for phishing URLs.

**Second-Level Domain (SLD):** SLD is the most critical part of a domain. Extracting features from SLD can help us get more information regarding the organization that registered the domain name. For example, in www.facebook.com, facebook is the SLD.

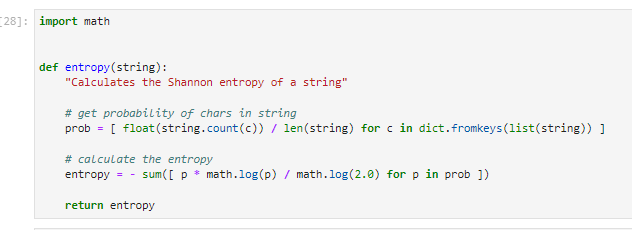
**Length:** The length of the domain consists of SLD and subdomains. Because phishing URLs or malicious domains are lengthy based on the previously discovered malicious URLs, considering the length feature is necessary.

**Numeric percentage:** This feature indicates the percentage of numerical characters to the length of the domain.



**Entropy:** This feature is based on the letter distribution and Shannon's entropy formula.





**N-gram:** This feature extracts the uni-gram, bi-gram, and tri-gram of the domain at the character level. ***(1-gram, 2-gram, 3-gram)***

**Distance from bad words:** There is a blacklist of all suspicious and harmful words. After tokenizing the domain to meaningful words, we get the average in distance of the domain's words from the blacklist.

**Obfuscation method:** To the best of our study, we considered nine different ways by which each domain can be obfuscated. If any of the following methods have been applied by the attacker, we mark the domain name as obfuscated:

1. **existing @ in Domain:**

This rule Using “@” symbol in the domain leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.

**Rule**: IF

1. **IP obfuscation Decimal 8 bits,**
2. **IP obfuscation Decimal 32 bits,**
3. **detecting IDN: suspicious domain name may be encoded with Unicode or international domain names encoded with Punycode, and**
4. **Shortened URLs: to confirm if it has been shortened or not, we send a request to see if it gets redirected to the real URL or not**
5. **Using Non-Standard Port :**

This feature is useful in validating if a particular service (e.g. HTTP) is up or down on a specific server. In the aim of controlling intrusions, it is much better to merely open ports that you need. Several firewalls, Proxy and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected. If all ports are open, phishers can run almost any service they want and as a result, user information is threatened. The most important ports and their preferred status are shown in Table 2. i.e ( ***port 80 and 443 are the only ports that should be open )***

We concluded the following rule to extract this future:

**Rule**: IF

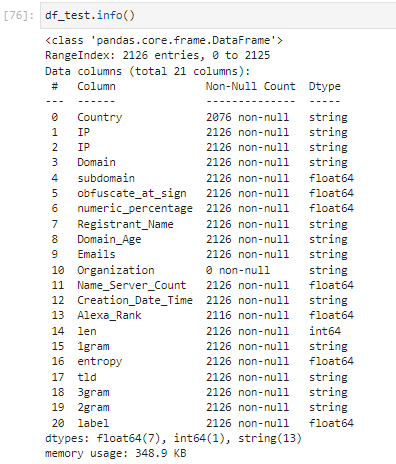
Table of Common ports to be checked

|  |  |  |  |
| --- | --- | --- | --- |
| ***PORT*** | ***Service*** | ***Meaning*** | ***Preferred Status*** |
| 21 | FTP | Transfer files from one host to another | Close |
| 22 | SSH | Secure File Transfer Protocol | Close |
| 23 | Telnet | provide a bidirectional interactive text-oriented communication | Close |
| 80 | HTTP | Hyper test transfer protocol | Open |
| 443 | HTTPS | Hypertext transfer protocol secured | Open |
| 445 | SMB | Providing shared access to files, printers, serial ports | Close |
| 1433 | MSSQL | Store and retrieve data as requested by other software applications | Close |
| 1521 | ORACLE | Access oracle database from web. | Close |
| 3306 | MySQL | Access MySQL database from web. | Close |
| 3389 | Remote Desktop | allow remote access and remote collaboration | Close |

1. ***Adding Prefix or Suffix Separated by (-) to the Domain:***

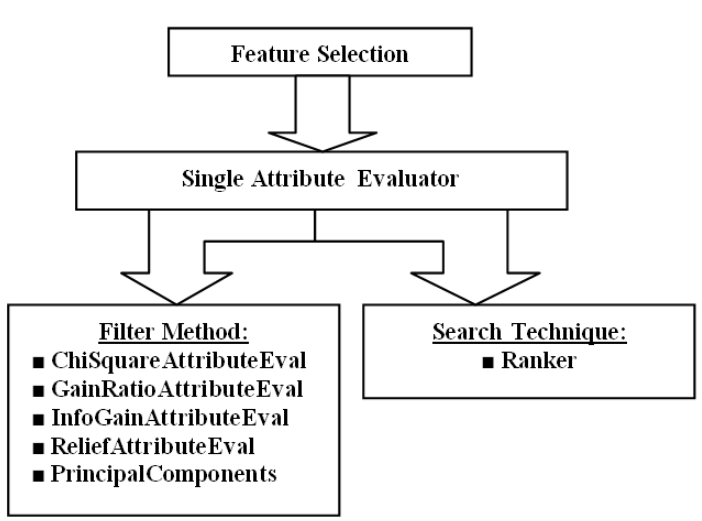
The researchers also concluded from the datasets in paper [13] that a dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example <http://www.Confirme-paypal.com/>. So they suggested the following rule:

**Rule**: IF



# Third party features

The third party features are extracted from two third party sources, i.e., Whois and Alexa rank and they contain the biographical properties of a domain. In [19] researchers have gone through feature evaluation using information gain analysis to get the merits of each feature in each category, applying the feature selection algorithm in Weka, i.e., ***InfoGainAttributeEval***, to obtain the top 13 features, out of the underlying dataset.



Proving the third party features as the most influential one among the top 13 features.

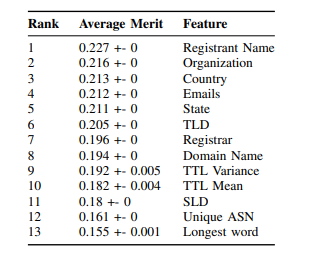


Figure 12 - FEATURE AVERAGE MERIT

#### **Domain Registration:**

Phishing website lives for a short period of time, so they suggest that trustworthy domains are regularly paid for several years in advance. In their dataset analysis, they found that the longest fraudulent domains have been used for one year only.

***Rule:*** IF

#### **Age of Domain:**

This feature can be extracted from WHOIS database (Whois 2005). ***Most phishing websites live for a short period of time.*** By reviewing the paper’s ***[14]*** dataset, the authors found that the minimum age of **the legitimate domain is 6 months**.

***Rule:*** IF

#### **DNS Record:**

For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records founded for the hostname (Pan and Ding 2006).

**If** the DNS record is **empty** or not found then the website is classified as “**Phishing**”,

***Otherwise*** it is classified as “**Legitimate**”.

***Rule***: IF

#### **Website Traffic:**

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, ***since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996)***.

By reviewing their dataset in article [13], they found that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore,

***if*** the **domain has no traffic or is not recognized by the Alexa database**, it is classified as “**Phishing**”.

***Otherwise***, it is classified as “**Suspicious**”

.

**Rule**: IF

#### **PageRank**

PageRank is a value ranging from “0” to “1”. PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage. In the datasets, ***they found that about 95% of phishing webpages have no PageRank.*** Moreover, the remaining 5% of phishing webpages may reach a PageRank value up to “0.2”.

***Rule:*** IF

**Table 1: List of the final 22 DNS-based features.**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Feature name** | **Description** |
| **Lexical** | | |
| **F1** | Subdomain | Has sub-domain or not |
| **F2** | TLD | Top-level domain |
| **F3** | SLD | Second-level domain |
| **F4** | Len | Length of domain and subdomain |
| **F5** | Numeric percentage | Counts the number of digits in domain and subdomain |
| **F6** | Entropy | Entropy of letter distribution |
| **F7** | 1-gram | 1-gram of the domain in letter level |
| **F8** | 2-gram | 2-gram of the domain in letter level |
| **F9** | 3-gram | 3-gram of the domain in letter level |
| **F10** | Distance from bad words | Computes average distance from bad words |
| **F11** | Obfuscation | Max value for URL obfuscation |
| **Third party** | | |
| **F12** | Domain name | Name of the domain |
| **F13** | Registrar | Registrar of the domain |
| **F14** | Registrant name | The name the domain has been registered |
| **F15** | Creation date time | The date and time the domain created |
| **F16** | Emails | The emails associated to a domain |
| **F17** | Domain age | The age of a domain |
| **F18** | Organization | What organization it is linked to |
| **F19** | State | The state the main branch is |
| **F20** | Country | The country the main branch is |
| **F21** | Name server count | The total number of name servers linked to the domain |
| **F22** | Alexa rank | The rank of the domain by Alexa |

# *Data Collection*

## Train Dataset: CIC-Bell-DNS 2021

In this research work [19] [18] they generated and released a large DNS features dataset of 400,000 benign and 13,011 malicious samples processed from a million benign and 51,453 known-malicious domains from publicly available datasets. The malicious samples span between three categories of spam, phishing, and malware. The dataset, namely CIC-Bell-DNS2021 replicates the real-world scenarios with frequent benign traffic and diverse malicious domain types.

One of the contributions of this research is to generate a large DNS features dataset based on the malicious and benign domains. As explained earlier, they sent HTTP requests to the gathered domains and the associated packets were captured. Then the captured packets were pre-processed, and the proposed features were extracted from the packets using a developed DNS-based feature extractor. Finally, the features were post-processed to create the final dataset. They managed to collect more than one million domains from various sources falling under four different categories of malware, spam, phishing, and benign domains. All the domains have been collected ***between May 2019 to June 2019*** and later updated with domains from **December 2020 for further validation**.

Each domain category is briefly explained as follows:

**Malware domains**

Malware category refers to the domains that have been previously identified to generate any general type of malware including drive-by download, DGA-based botnets, Distributed Denial of Service (DDoS) attacks, and spyware. The malware domains were collected from DNS-BH [19] and malware domain list [20].

**Spam domains**

Spammers employ different ways to find valid email addresses for sending bulk emails. Dictionary harvest attack is one of the common ways to seek a valid email address by randomly sending mail to widely used mailbox names for a domain.

**Phishing domains**

Phishing domains imitate the looks of legitimate websites and leverage social engineering techniques to trick users into clicking the malicious link. Upon clicking the fake link through email or SMS, the user is directed to an imposter website asking for the user's login credentials and private information. In the dataset, the phishing domains were collected from OpenPhish [17] and PhishTank [18] which are collaborative phishing verification websites where the users submit the phishing data, and the community users vote for it.

**Benign domains**

All the domains that are not in the above categories are deemed to be benign. They gathered all benign domains from Majestic Million [16]. Table below shows the breakdown of the dataset in terms of the number of collected domains (original domain) in each category of malware, spam, phishing, and benign. After sending HTTP requests to each domain, some of the domains do not respond, e.g., C&C servers that are not alive anymore. The remaining domains that **respond OK (200)** are logged and, associated DNS packets are dumped. In Table below, the domains and packets that have been successfully processed are identified with columns “domains processed” and “packets processed”, respectively. [18] *As four our research project we only took phishing and benign CSV files and combined them to generate a training datasets.*

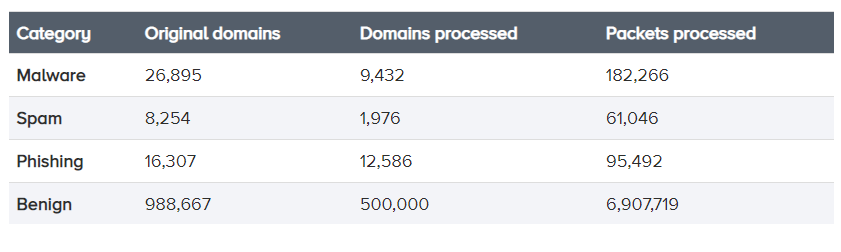


Figure 13 - Statistics of the domains dataset

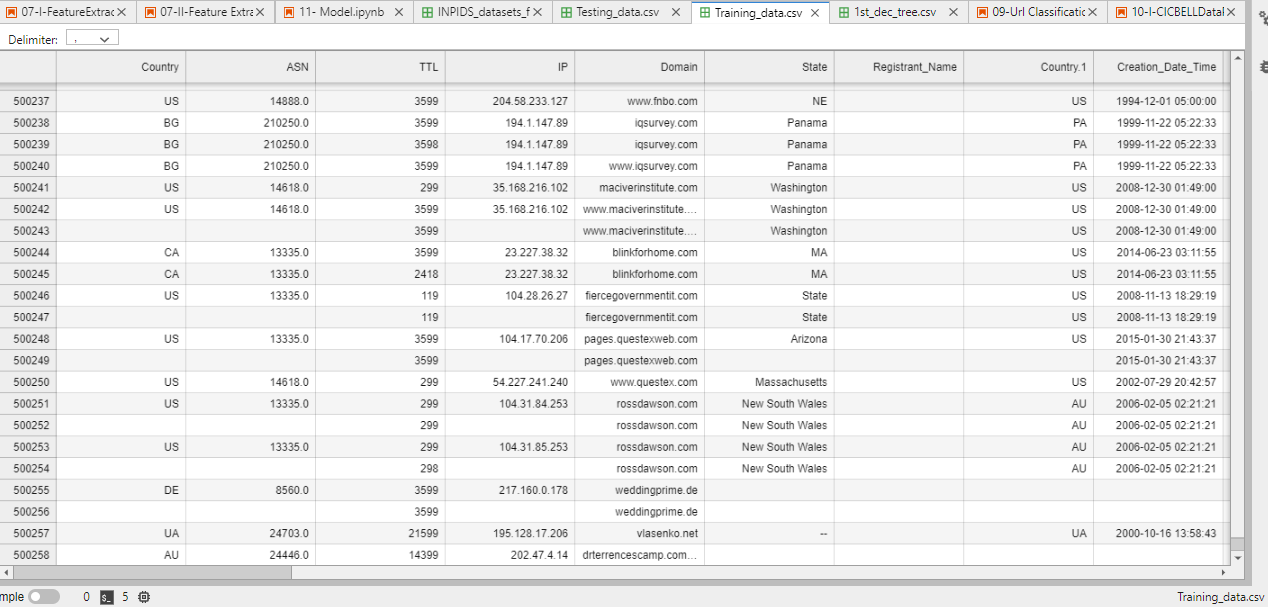


Figure 14- Processed Training Dataset CSV file contains 5012 586 row of benign and phishing Domains and their features

## Test Dataset: real time based scenario

**Fig.15.** describes all the steps involved in the test dataset generation process which will play the role of real time packet transfer scenario; Starting with downloading the URLs data CSV files containing phishing and legitimate URLs, then transferring it into a PCAP file using a python sniffer called Passer that can work off a live packet capture. The next step is extracting HTTP and DNS lookup logs from this PCAP file so we can enrich the sets with other columns that will be useful in the feature extraction phase (same 22 DNS-based features as the CIC BELL training dataset) and finally fitting it into a Machine Learning model.

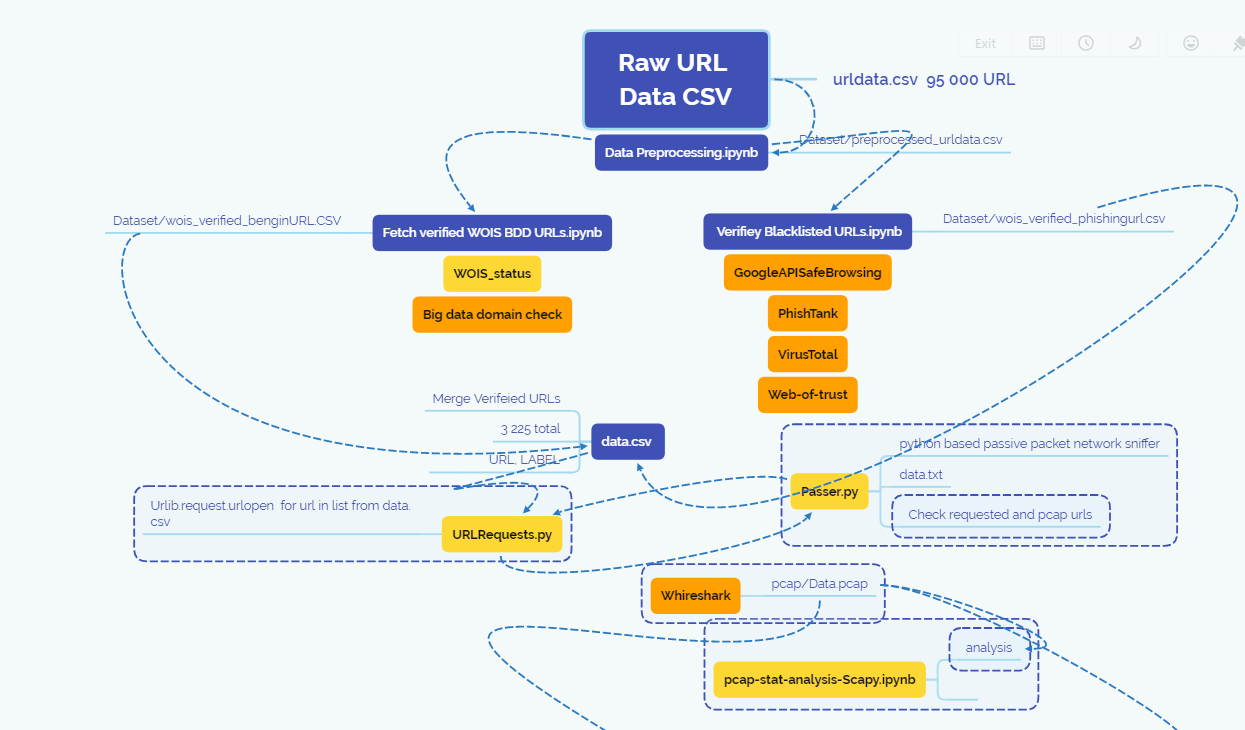




Figure 15- Design for generating synthetic test datasets that represent a real time packet flow scenario

Based on the research design in **figure 15.** And the sections these are the major steps involved in the data collection phase for the testing data:

**Data Preprocessing:** removed nulls and duplicate records since they affect our Machine Learning (ML) model. Also, removed unnecessary columns and retained only URL and Label columns.

**WOIS Verification:** the preprocessed data was split into benign sample set and malicious sample set. These two sample sets were individually checked for reachability of Whois to ensure our features data is not filled with meaningless features. I also checked from blacklist sites and phish tank if the Urls are malicious or begins to avoid throwing ML model off balance as well as Big Data domain.

**Verified CSV file:** the Whois and black site API verified benign and malicious URLs were further sampled to 5000 URLs each and merged to get a final dataset for extracting features in ***data.csv*** with 10 000 Urls.

**Generating PCAP file:** I first had to create a python code that can insert the 10 000 Urls from the *data.csv* file into a list then with a FOR loop I open connexion to each URL.



Figure 16- URLibRequest.py source code to open connexion to the List of URLs

Then I launched the python3 code in a kali lunix terminal to open the connexions.

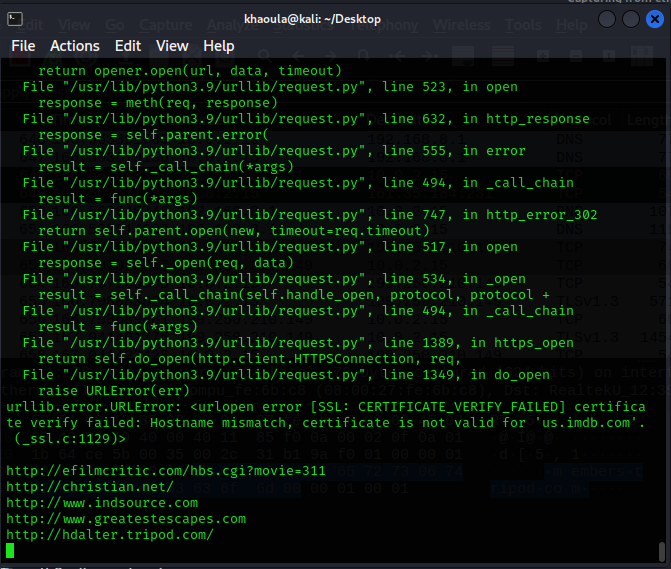


Figure 17- Outputs of the URLibrequet.py in kali lunix terminal

Using Wireshark in Kali Linux and a python sniffer called Passer that can work off a live packet capture from the other python code I launched in terminal 1, then generated the PCAP file.

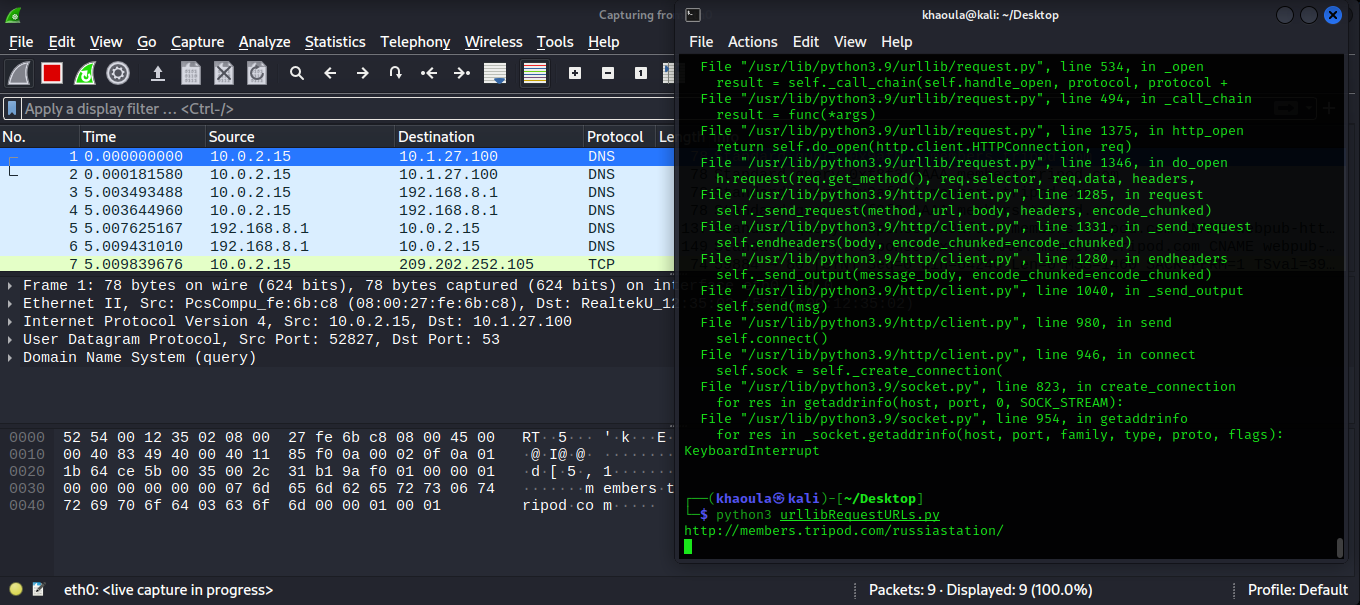
****

Figure 18- The outputs of Wireshark and passer.py in terminal 2

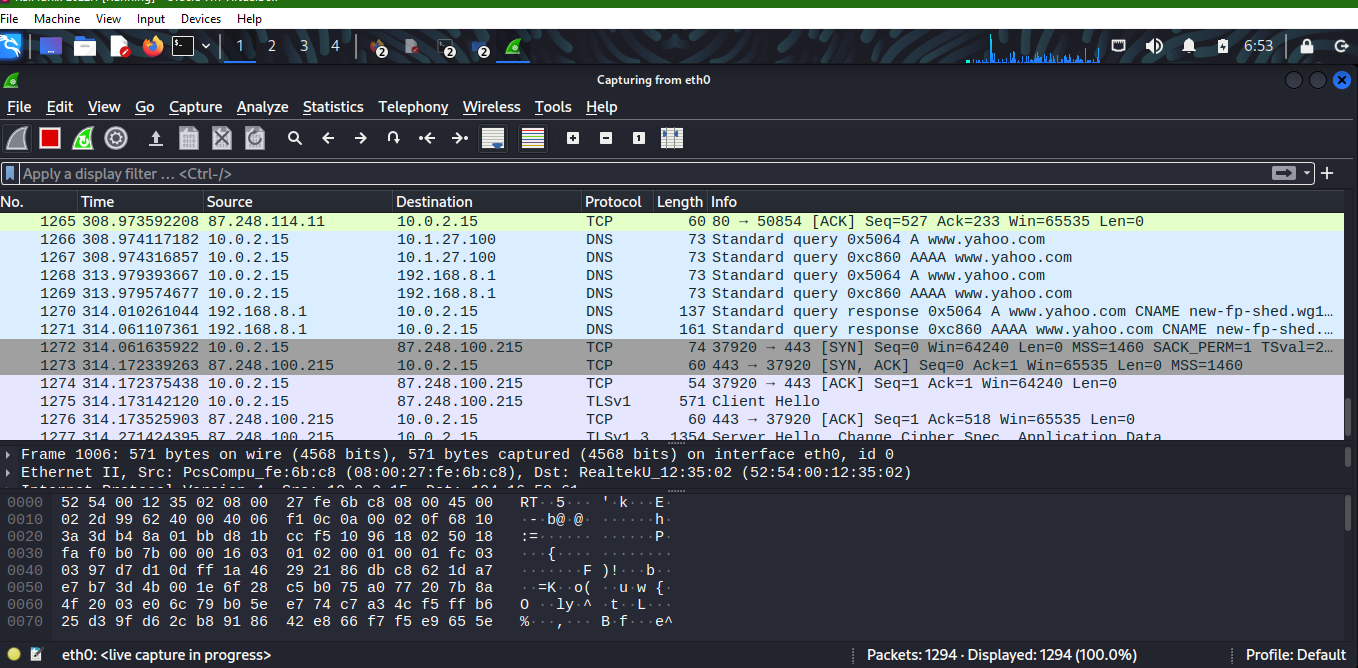


Figure 19- Wireshark Packet capture outputs of the 10 000 url connexions

The outputs of both terminals can be viewed in the following Screenshot:

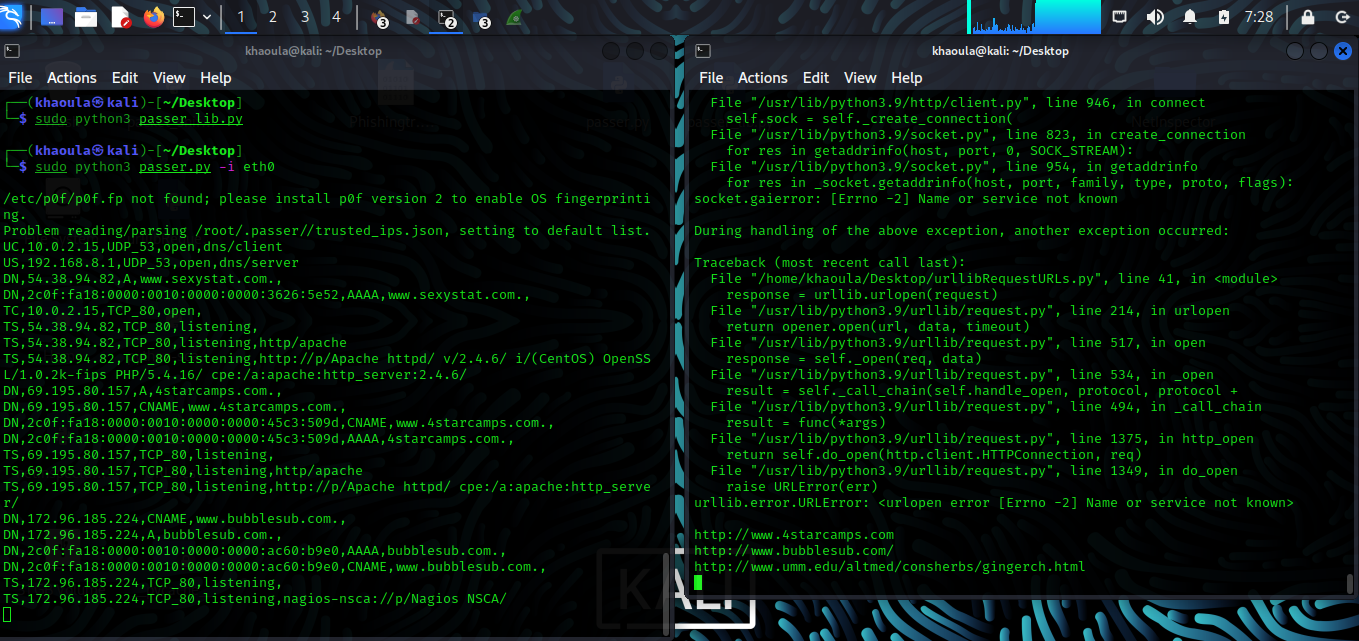
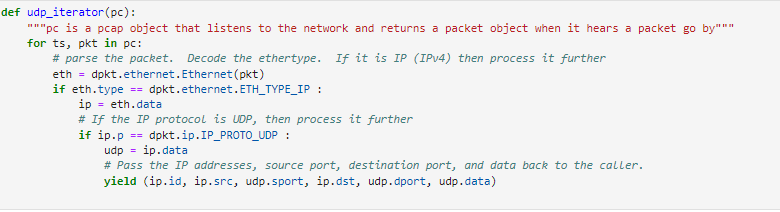
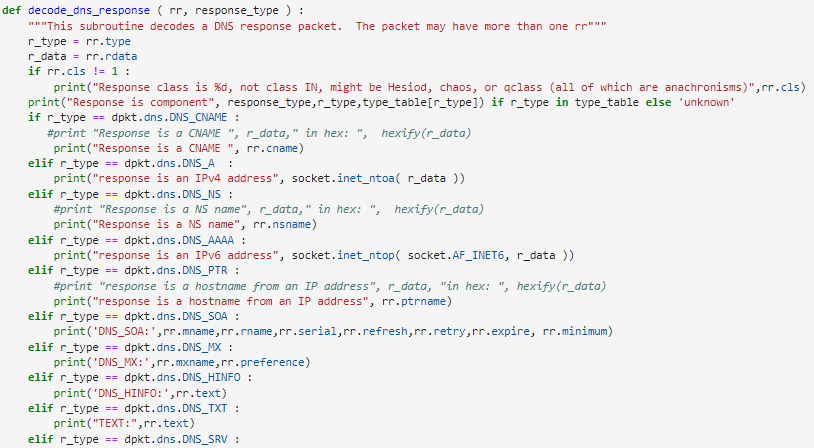


Figure 20- terminals outputs for passer.py and urlibrequest.py

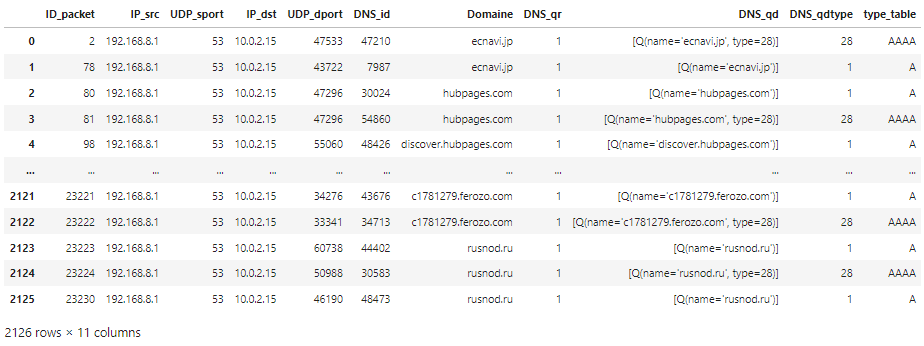
**Generating DNS and HTTP logs:** using **SCAPY and DPKT** libraries in python to pars a PCAP file and the packets within it.

I created a tool that reads the raw PCAP file, parses and decodes the Ethernet, IP, TCP, and HTTP layers, and prints out the URI of the HTTP requests by using a PCAP object that listens to the network and returns a packet object when it hears a packet go by.





I generated *DNS\_logs.csv* and *HTTP\_logs.csv* that will be helpful in extracting features.

**

**Enriching the Testing dataset:** after fusing the http and DNS logs into a single data frame I used pre-defined functions to enrich the datasets. At the end of this step we get our final *testing\_data.csv* file that will be used to extract features and train our model to classify phishing attacks.

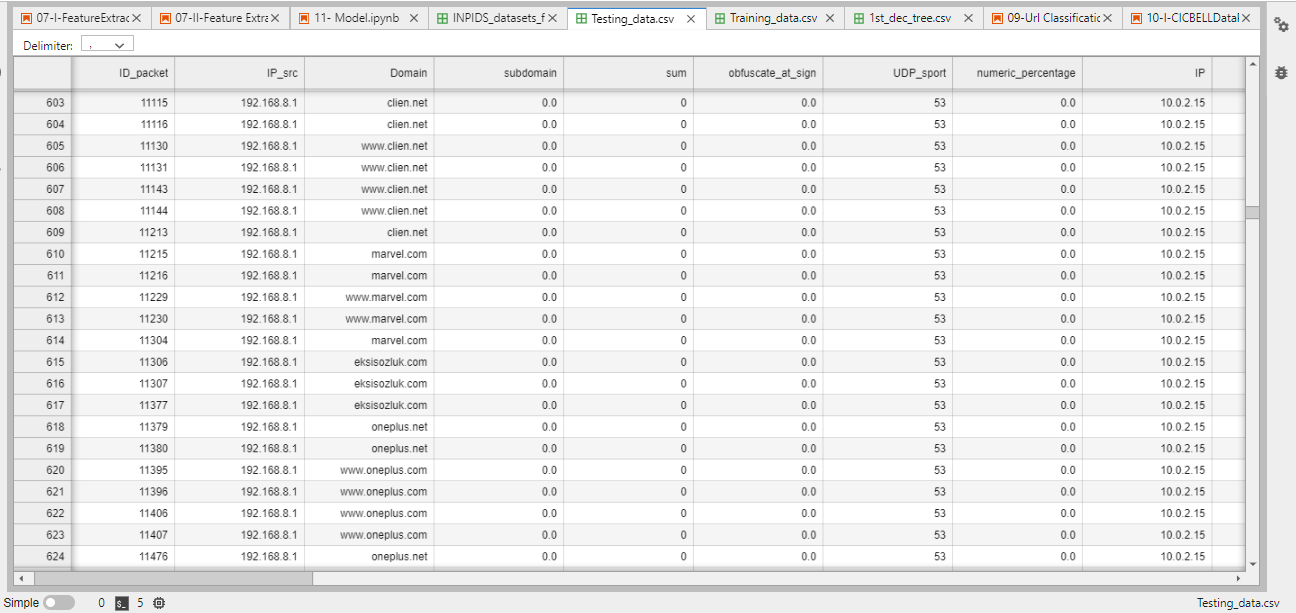
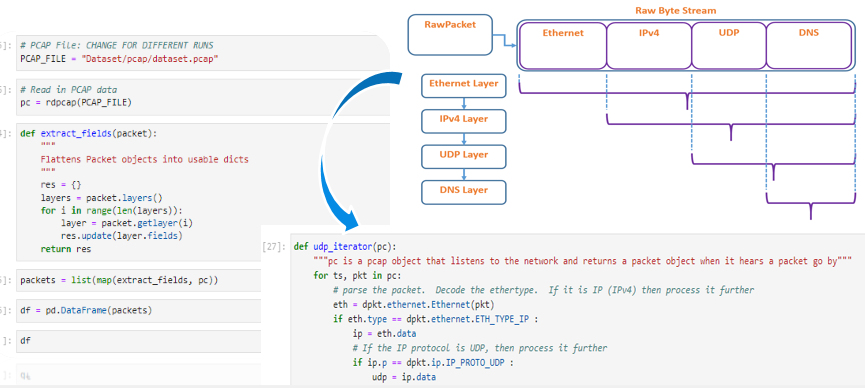


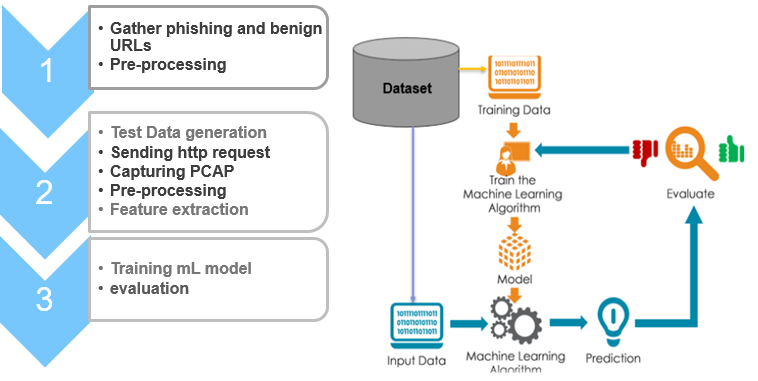
Figure 21 - Resulted Testing Dataset with 2126 entries for benign and phishing Domain names and its 22 DNS-based extracted features

# *Models Implementation*

We provide a methodology for feature engineering of packet captures. We developed 22 ***(currently)*** clearly defined discriminative features including lexical-based and third party-based (biographical) features based on the CIC BELL dataset [18]. First, the captured DNS PCAP file is read and all the domains in the answer section of type A, AAAA, and CNAME query responses are retained. The field “rrname” keeps the domain name. Meanwhile, the statistical features are extracted from the structure of the DNS message in a specific packet window. Then, for each captured domain, we extract the lexical and third party features.



The categorical features namely, sld, emails, domain name, country, registrar, state, registrant name, longest word, organization, tld are transformed and represented as continuous values. Average of n-gram frequencies, typos, and distance from bad words were computed. Also, creation date time and domain age are converted to seconds and years, respectively. ***The rest of the numerical features remain unchanged.*** Besides, we calculate the maximum value of the nine obfuscation methods and merge them all into ***one feature of obfuscation***. Finally, we create a CSV file of all the post-processed features values. The third stage trains the ML model, and the fourth stage validates and evaluates the model using fold cross-validation.



We usedmultiple machine learning algorithms namely:

1. Decision Tree

A decision tree is a supervised machine learning algorithm that can be used for both classification and regression problems. A decision tree is simply a series of sequential decisions made to reach a specific result. [21]

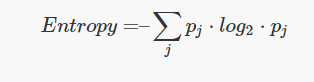


We Created the classifier, fit it on the training data and made predictions on the test set, we choosed **max\_depth=3** as the depth of the tree known as a hyper parameter before we fit the model to the data.

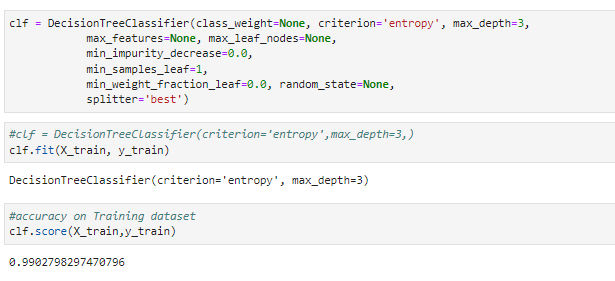
Because my decision boundary is too complex, I can over fit the data, which means that my model will be describing noise as well as signal.

If the **max**\_**depth** is too small, we might be under fitting the data, meaning that the model doesn't contain enough of the signal.

In the decision tree Python implementation of the scikit-learn library, this is made by the parameter ‘criterion‘. This parameter is the function used to measure the quality of a split and it allows users to choose between ‘Gini ‘or ‘entropy‘. The entropy is calculated using the following formula:

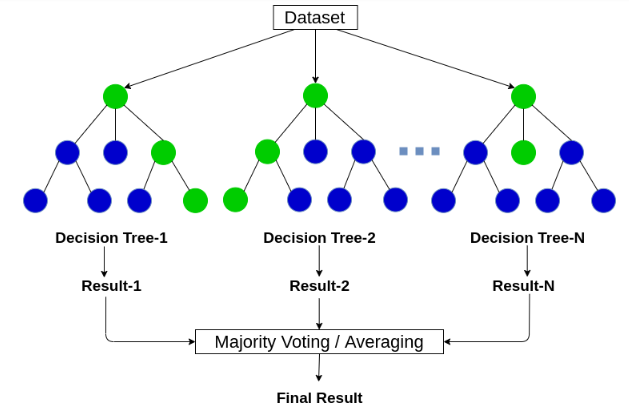


Entropy is a measure of information that indicates the disorder of the features with the target. Similar to the Gini Index, the optimum split is chosen by the feature with less entropy. It gets its maximum value when the probability of the two classes is the same and a node is pure when the entropy has its minimum value, which is 0.



1. Random Forest

Random Forest is a tree-based machine learning algorithm that leverages the power of multiple decision trees for making decisions. It is a forest of **randomly created decision trees.** Each node in the decision tree works on a random subset of features to calculate the output. The random forest then combines the output of individual decision trees to generate the final output. [21]



"Tree Ensembles" or Random Forest often have the best predictive accuracy with labeled, tabular data because trees can fit non-linear, non-monotonic relationships, and interactions between features.

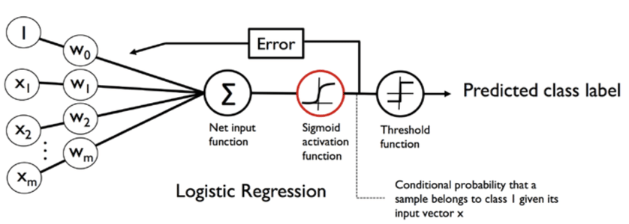
A single decision tree, grown to unlimited depth, will over fit. We solved this problem by assembling trees, with bagging (Random Forest) because the latter may be less sensitive to hyper parameters. We also tested with not only One-hot encoding of categorical features for tree ensembles but also with arbitrary "ordinal" encoding (Randomly assigning an integer to each category.) Compared to one-hot encoding, the dimensionality was lower, and the predictive accuracy was the same and even slightly better.



***In future work, to*** convert the categorical features into numeric attributes because we have variables with a high number of categorical levels, we are considering combining levels or using ***the hashing trick.***

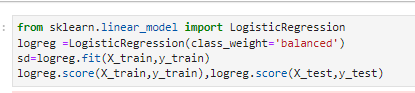
1. Logistic Regression

**Logistic regression** is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. [22]



 class\_weight of LR works by penalizing mistakes in samples of class[i] with class\_weight[i] instead of 1. So higher class-weight means you want to put more emphasis on a class. From what you say it seems class 0 is 19 times more frequent than class 1. So you should increase the class\_weight of class 1 relative to class 0, say {0:.1, 1:.9}. If the class\_weight doesn't sum to 1, it will basically change the regularization parameter.

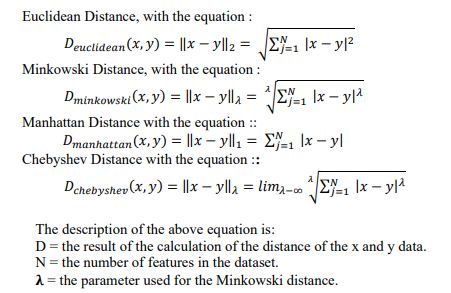
For our model version we used class\_weight="balanced" it basically means replicating the smaller class until we have as many samples as in the larger one, but in an implicit way.



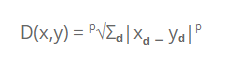
1. K Nearest Neighbor (KNN)

KNN is an algorithm, based on the local minimum of the target function which is used to learn an unknown function of desired precision and accuracy. The algorithm also finds the neighborhood of an unknown input, its range or distance from it, and other parameters. It’s based on the principle of “information gain”—the algorithm finds out which is most suitable to predict an unknown value.  [23]

In KNN, there are a few hyper-parameters that we need to tune to get an optimal result. Among the various hyper-parameters that can be tuned to make the KNN algorithm more effective and reliable, the distance metric is one of the important ones through which we calculate the distance between the data points as for some applications certain distance metrics are more effective. There are many kinds of distance functions that can be used in KNN such as Euclidean Distance, Hamming distance, Minkowski distance, Kullback-Leiber (KL) divergence, BM25 etc.



Euclidean distance can be generalized using **Minkowski** norm also known as the p norm. The formula for Minkowski distance is:

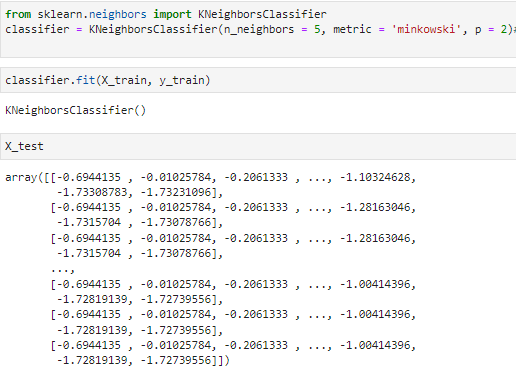


Here we can see that the formula differs from the formula of Euclidean distance as we can see that instead of squaring the difference, we have raised the difference to the power of p and have also taken the p root of the difference. Now the biggest advantage of using such a distance metric is that we can change the value of p to get different types of distance metrics.

**p = 2**

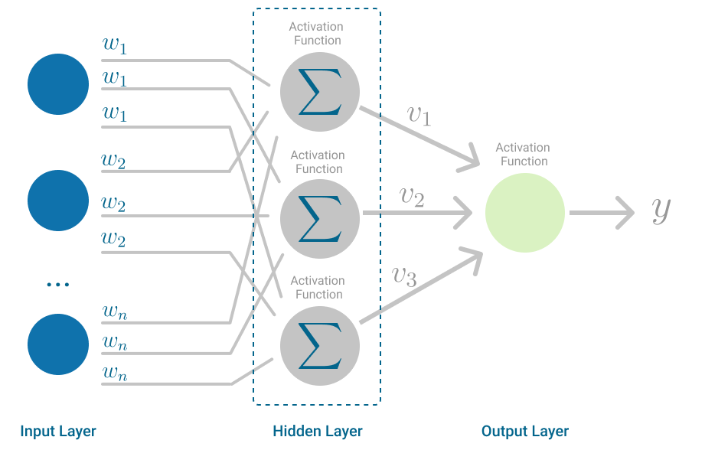
If we take the value of p as 2 then we get the Euclidean distance. We also chooses to go with 5 as a number of neighbors to use by default for [**kneighbors**](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.kneighbors) queries.

.



1. Multilayer Perceptron

Multilayer Perceptron has input and output layers, and one or more **hidden layers** with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function. [24]



We created the MLP and activated it by means of ReLU so we can funnel the information into a very dense format. This way, the model will be capable of learning the most important patterns, which helps generalizing to new data.

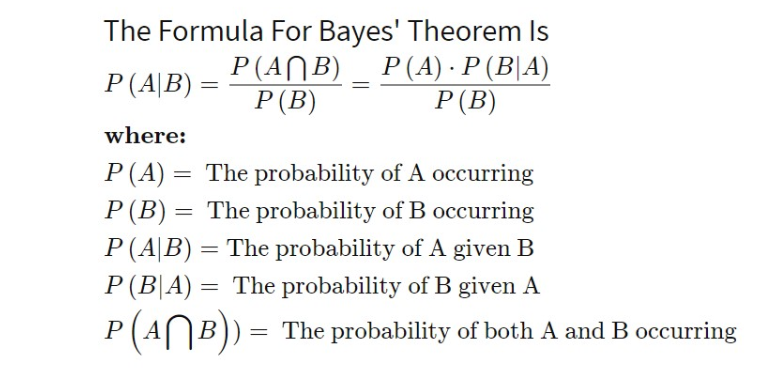
Finally, there's an output layer, which has num\_classes output neurons and activates by means of Softmax. The number of neurons equals the number of scalars in our output vector. Since that data must be categorical for categorical cross entropy, and thus the number of scalar values in our target vector equals the number of classes, it makes sense why num\_classes is used. Softmax, the activation function, is capable of generating a so-called multiclass probability distribution. That is, it computes the probability that a certain feature vector belongs to one class. We just configured the model architecture.

So the model can learn we configure the following hyper parameters:

We use categorical cross entropy as our loss function and the Adam optimizer for optimizing our model. It combines various improvements to traditional stochastic gradient descent (Kingman and Ba, 2014; Ruder, 2016). Adam is the standard optimizer used today .Accuracy is highly intuitive to humans so we'll use that alongside our categorical cross entropy loss. Next, we fit the training data to our model. We choose 10 epochs, or the number of iterations before it stops training, a batch size of 250, verbosity mode 1 and a validation split of 20%.

1. Gaussian Naive Bayes

Naive Bayes Classifiers are based on the Bayes Theorem. One assumption taken is the strong independence assumptions between the features. These classifiers assume that the value of a particular feature is independent of the value of any other feature. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently. [25]

****

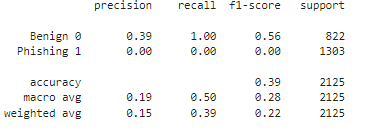
# *Preliminary Results and Analysis*

1. Numerical Features

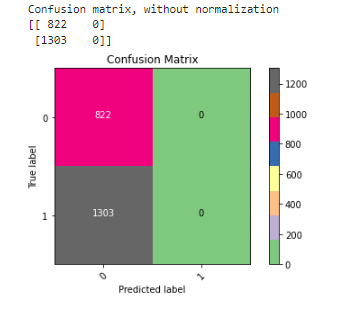
We train and validate the classification models focusing on numerical features in the first analysis:

7.1 Decision Tree

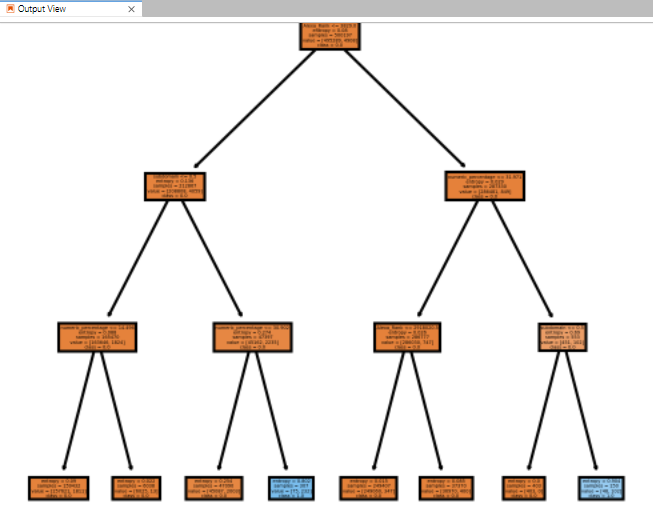
**Classification report:**



**Confusion Matrix:**

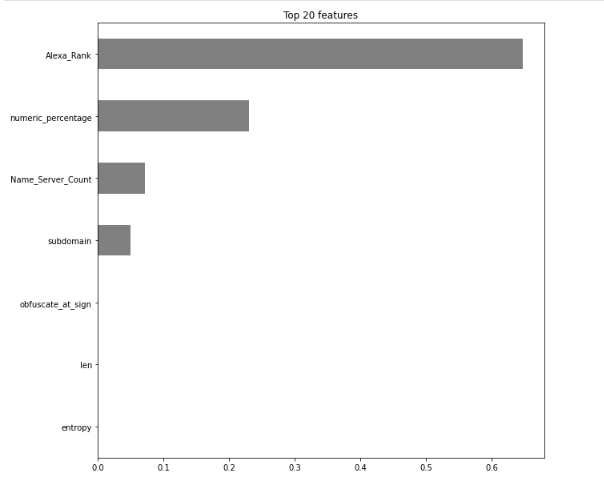


**Decision Tree Output view:**

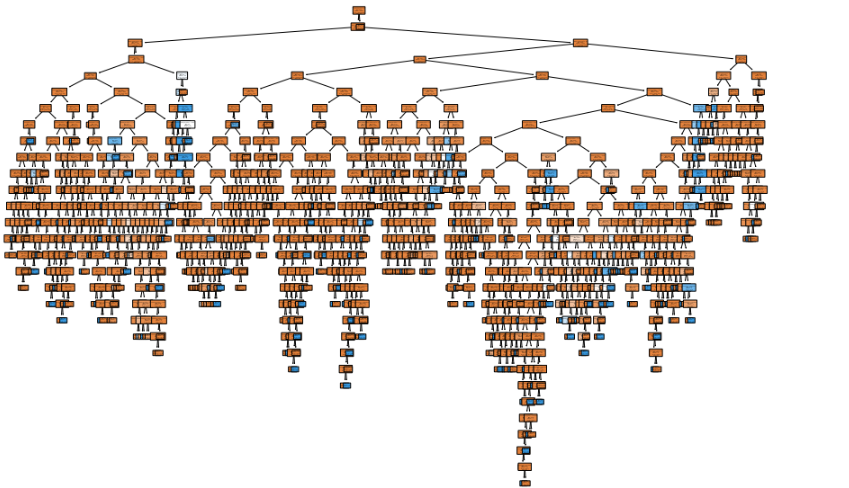


7.2 Random Forest

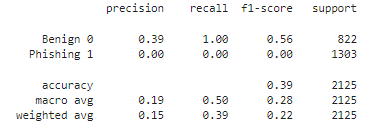
**Top 20 numeric features table:**



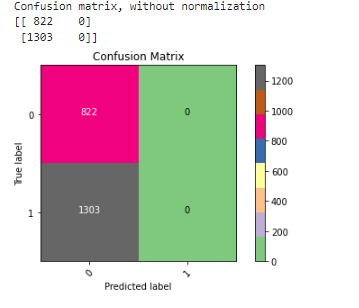
**Random forest Output view:**



**Classification report:**

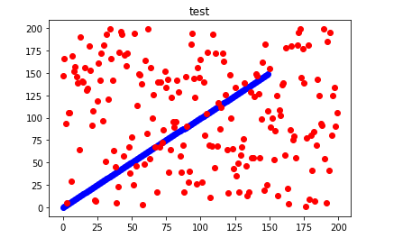


# **Confusion Matrix:**

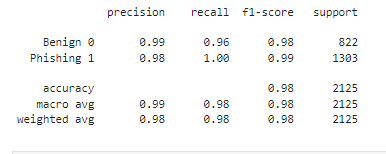


* 1. Logistic regression

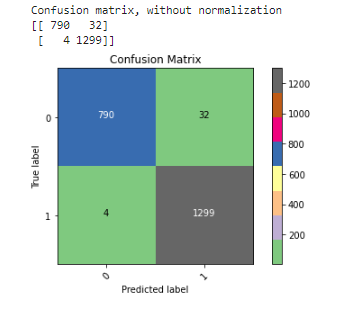
**Visualizing the y\_test, y\_pred results:**



**Classification report:**

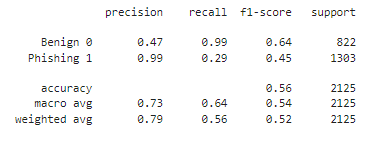


# **Confusion Matrix:**

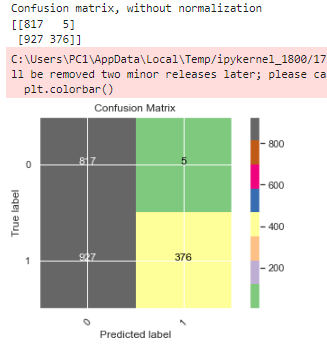


* 1. KNN

**Classification report:**

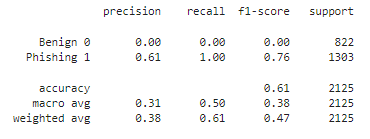


# **Confusion Matrix:**

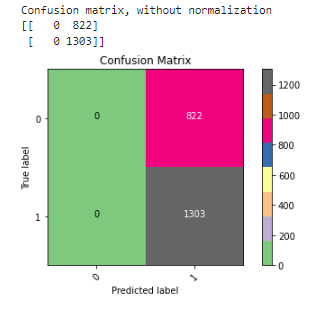


* 1. MLP

**Classification report:**

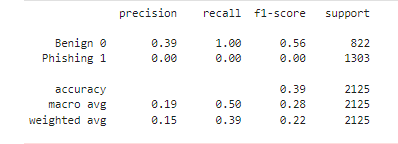


**Confusion Matrix:**

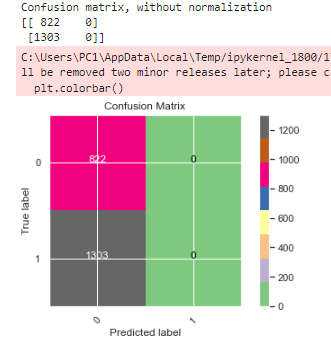


* 1. GNB

**Classification report:**



**Confusion Matrix:**



***In the analysis step,*** we applied a set of machine learning algorithms on two subdivisions of the dataset, namely balanced and imbalanced.

We compare the classification results of multiple machine learning algorithms as seen in pervious sections showing the classification report and confusion matrix of each model. We used stratified 5-fold cross validation from the sklearn library which returns stratified folds for training and testing data by preserving the percentage of samples for each class. We shuffle each class’s samples before splitting into train and test bins. As illustrated Logistic regression (LR) significantly outperforms k-NN, MLP, GNB,RF,DT and in terms of accuracy, f1-score, and precision It achieves 98% and 99% For 97/3% data ratio, the gaps between classification results are trivial where Multilayer perceptron MLP ranks second among other classifiers with a notable difference (almost 37% lower for accuracy) from LR. Overall, the results on the imbalanced dataset are superior to ones on the balanced dataset due to a higher ratio of benign samples that might influence the average classification measures. However, since we have used stratified sampling, we can make sure we have provided more uniformity during training by removing the variance of the proportions inside bins.

1. Categorical and numerical Features

We train and validate the classification models focusing now on both encoded categorical and numerical features in the second analysis:

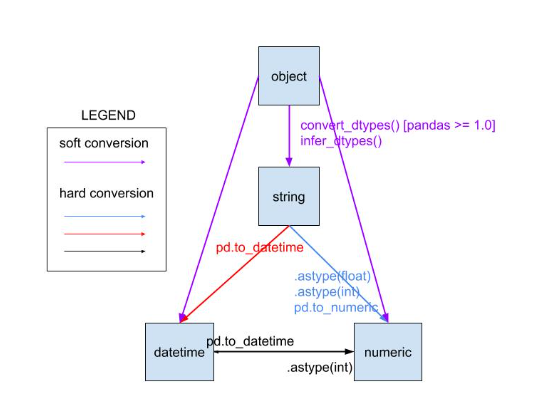
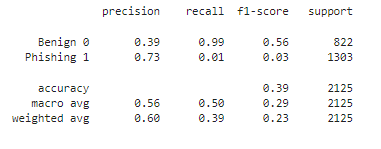


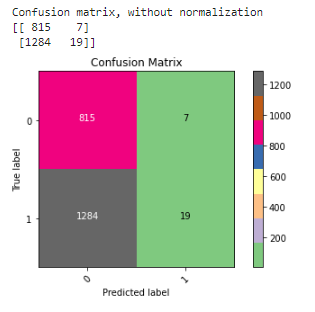
Figure 22 - Soft and Hard conversions of categorical features used in our models next to One-hot encoder

* 1. Decision Tree

**Classification report:**

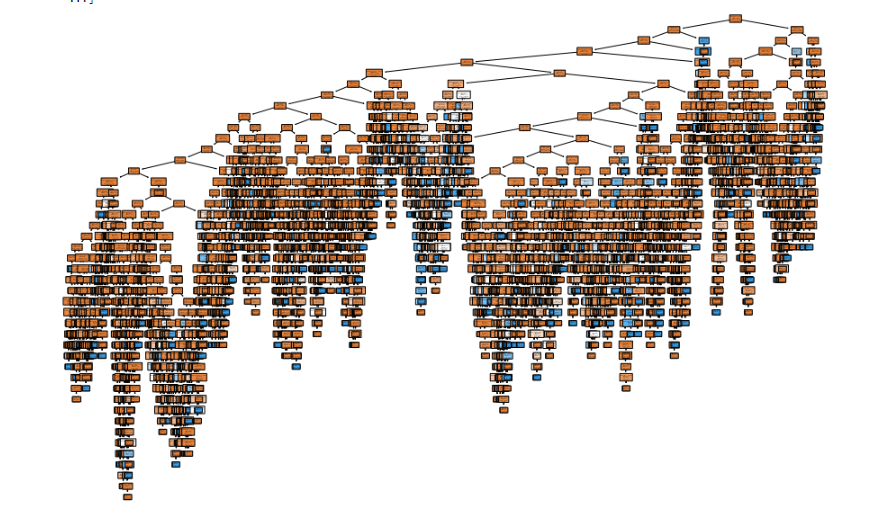


**Confusion Matrix:**

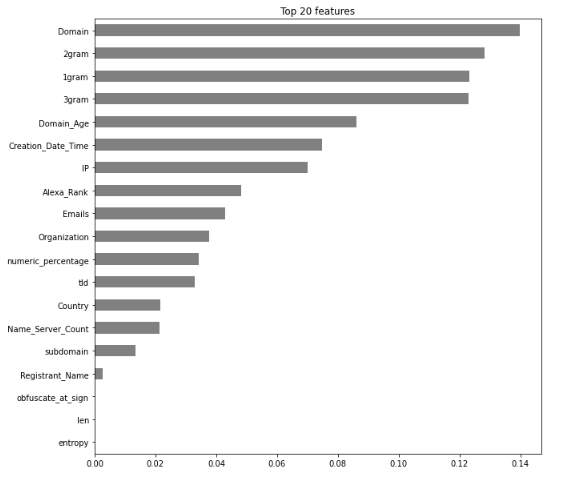


* 1. Random Forest

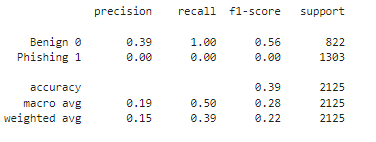
**Random forest Output view:**



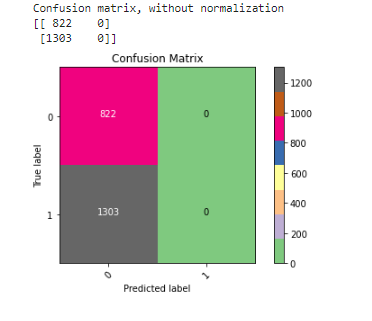
**Top 20 features table:**



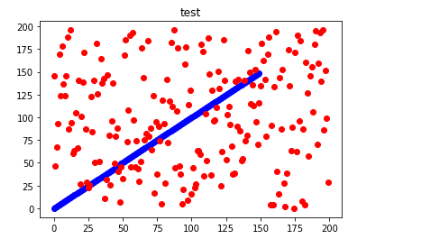
**Classification report:**



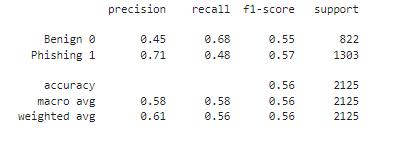
**Confusion Matrix:**



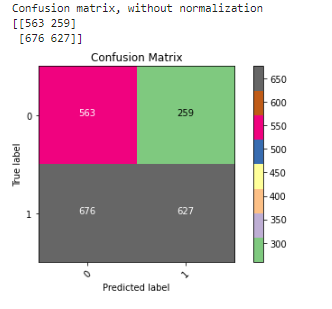
8.3 Logistic regression



**Classification report:**

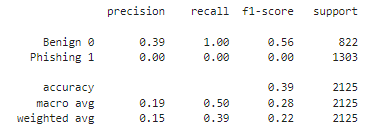


**Confusion Matrix:**

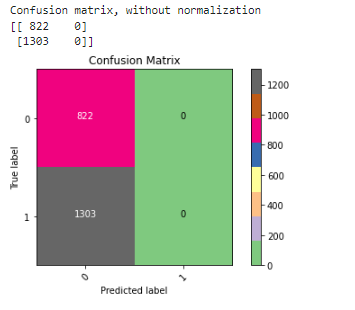


* 1. KNN

**Classification report:**

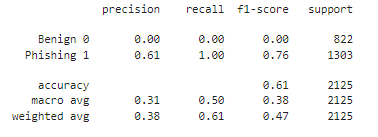


**Confusion Matrix:**

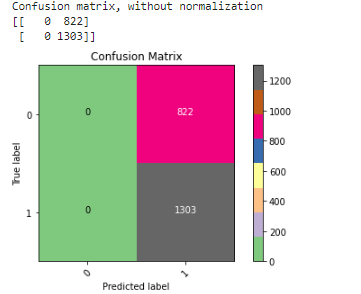


* 1. MLP

**Classification report:**

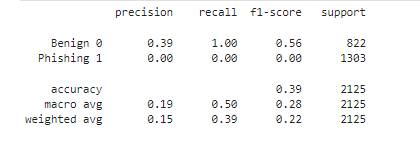


**Confusion Matrix:**



* 1. GNB

**Classification report:**



**Confusion Matrix:**

# 

***In the second analysis,*** we compare the classification results of multiple machine learning algorithms focusing now on both encoded categorical and numerical features as seen in pervious sections showing the classification report and confusion matrix of each model. As illustrated , now Multilayer perceptron (MLP) significantly outperforms k-NN, LR, GNB,RF,DT in terms of accuracy, f1-score, and precision It achieves 61% and 76% For 97/3% data ratio, the gaps between classification results are trivial where Logistic regression (LR) ranks second among other classifiers with a notable difference (almost 05 % lower for accuracy) from MLP.

Overall, we noticed the results on the numeric features are superior to the ones with encoded categorical features because we need to find the right distance function that works for our dataset. The use of binary indicator variables solves this problem implicitly. This has the benefit of allowing us to continue our probably matrix based implementation with this kind of data, but a much simpler way - and appropriate for most distance based methods - is to just use a modified distance function which will be our future work to increase the accuracy rate.

There is an infinite number of such combinations. We need to experiment which works best for our case. Essentially, we might want to use some classic metric on the numeric values (usually with normalization applied; but it may make sense to also move this normalization into the distance function), plus a distance on the other attributes, scaled appropriately.

In most real application domains of distance based algorithms, this is the most difficult part, optimizing our domain specific distance function.

# *Conclusion and* Future Work

In this research, we proposed an applicable phishing domain classification detection system based on lexical and third-party features acquired by deep inspection of DNS traffic. The proposed detection system achieves a somewhat promising preliminary results on several machine learning techniques. We also have shown that third-party features are the most important category of features in the classification process with a total 58% information gain among the top 13 features. We have worked on implementing a passive python network sniffer that capture live flow traffic and save it to a PCAP file so we can extract 22 features from 500,000 benign and 5,011 malicious DNS responses captured from over seven million DNS packets. In the future, we are planning to enrich our feature set by adding more state full /stateless features to add up to 35 features *as well as orienting this INPDS system towards HTTPS protocol.*

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough. The authors in (Mohammad, Thabtah and McCluskey 2012) **[14]** (Mohammad, Thabtah and McCluskey 2013) **[15]** suggest checking the certificate assigned with HTTPS including the extent of the trust certificate issuer, and the certificate age. Certificate Authorities that are consistently listed among the top trustworthy names include: “GeoTrust, [GoDaddy](http://www.godaddy.com/gdshop/ssl/ssl.asp?isc=BESTSSL1), Network Solutions, Thawte, Comodo, Doster and VeriSign”. Furthermore, by testing out the datasets in paper **[13]**, they found that the minimum age of a reputable certificate is two years.

***Rule:*** IF

And like the model approach in **[16]** we want to detect malicious HTTPS traffic without decryption in future work. To detect malicious HTTPS traffic, the authors analysed the different log files generated by ***Bro IDS (Now renamed Zeek)*** which provide comprehensive information about HTTPS traffic. They also used XGBoost machine learning algorithm to classify the traffic. The features are based on three different log files: - conn.log, ssl.log, and x509.log. They define their feature model by analysing the logs generated by Bro/Zeek; these logs provide detailed information about HTTPS traffic.

In general Bro generates different kinds of log files as follows:

➢ **conn.log:** TCP/UDP/ICMP connections

➢ **ssl.log:** A record of SSL sessions, including certificates being used.

➢ **x509.log:** X.509 certificats information.

➢ **http.log:** All HTTP requests with their replies.

➢ **Smtp.log:** a description of SMTP activity.

➢ **ftp.log:** A log of FTP session activity.

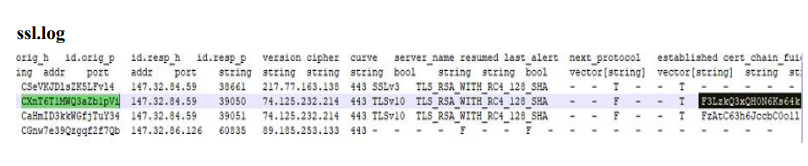
➢ **dns.log:** DNS queries with their responses.

➢ **dpd.log:** A summary of protocols used on non-standard ports.

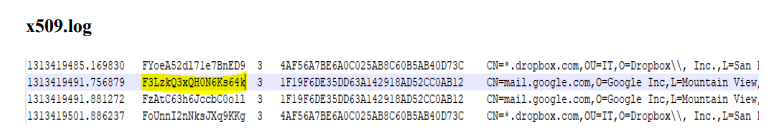
➢ **files.log:** Description of files transferred over the network and all the information collected from different protocols.

For the next step we want to extract the following:

***1. SSL record:*** The **ssl.log** file provides information about SSL/TLS handshakes and encryption establishment process. It contains versions of SSL/TLS, server names, ciphers used, certificate path, subjects and issuer.



***2. Certificate record:*** The **x509.log** file contains the certificate record describing the information about certificate, such as certificate serial number, common names, time validities, subjects, certificate key length, etc



**3**. **Validation period of the certificates:** The validity of a certificate can be checked by using the capture time and the validity period of the certificate. If the capture time is within the certificate validity period, then it is valid. The datatype of this feature (𝑅8) is Boolean. If the certificate is valid, then the feature value is true, and the feature value is false if the certificate is not valid.

**4.** **Number of domains in certificate SAN DNS**: The SAN (Subject Alternative Names) describes which domains belong to this certificate. For example, a Google certificate SAN DNS is {\*.google.com, \*. google.ca, \*. google.co.in, \*. google.cl, \*. google.co.uk, \*. google.de}. For each new incoming certificate, the number of DNS in SAN is stored in a list. The average is calculated from the list.

In conclusion, the aim of the future research is to examine the HTTPS traffic PCAP file by extracting the elements previously highlighted in this work as well as other studies and deciding whether it is malicious or not. Connection information, SSL records, and certificate records will be used to extract the features. The connection will then be established depending on the source IP, destination IP, destination ports, and protocol. The output is labelled as either suspicious or normal. **This stage is still in progress** since we are presently working on HTTP traffic to figure out how we can identify the most critical **features that will not be encrypted during transfer.**

# Bibliographic references

***[1] - Keyfactor. (2021, October 11). HTTPS Phishing Attacks: How Hackers Use SSL Certificates to Feign Trust –.*** [***https://www.keyfactor.com/blog/https-phishing-attacks-how-hackers-use-ssl-certificates-to-feign-trust/***](https://www.keyfactor.com/blog/https-phishing-attacks-how-hackers-use-ssl-certificates-to-feign-trust/)

***[2] -***  ***Detection modules — Slips 0.8.4 documentation. (2020). STRATOSPHERE.*** [***https://stratospherelinuxips.readthedocs.io/en/develop/detection\_modules.html#***](https://stratospherelinuxips.readthedocs.io/en/develop/detection_modules.html)

***[3] - Get. (2022, January 25). Zeek.*** [***https://zeek.org/get-zeek/***](https://zeek.org/get-zeek/)

***[4] - Detection of Encrypted Malicious Network Traffic using Machine Learning. (2019, November 1). IEEE Conference Publication | IEEE Xplore.*** [***https://ieeexplore.ieee.org/document/9020856***](https://ieeexplore.ieee.org/document/9020856)

***[5] - Nohe, P. (2019, May 22). 58% of Phishing Websites Now Use HTTPS. Hashed Out by The SSL StoreTM.*** [***https://www.thesslstore.com/blog/58-of-phishing-websites-now-use-https/***](https://www.thesslstore.com/blog/58-of-phishing-websites-now-use-https/)

***[6] - Le, H. (2018, February 9). URLNet: Learning a URL Representation with Deep Learning for. . . arXiv.Org.*** [***https://arxiv.org/abs/1802.03162***](https://arxiv.org/abs/1802.03162)

***[7] – Xiang, G. (2011). [PDF] CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites | Semantic Scholar. Semantic Scholar.*** [***https://www.semanticscholar.org/paper/CANTINA%2B%3A-A-Feature-Rich-Machine-Learning-Framework-Xiang-Hong/eca05441703a2913b45130d7c33ea51090cb5e3d***](https://www.semanticscholar.org/paper/CANTINA%2B%3A-A-Feature-Rich-Machine-Learning-Framework-Xiang-Hong/eca05441703a2913b45130d7c33ea51090cb5e3d)

***[8] – Lee, Y. (2020). [PDF] CATBERT: Context-Aware Tiny BERT for Detecting Social Engineering Emails | Semantic Scholar. Semanticscholar.Org.*** [***https://www.semanticscholar.org/paper/CATBERT%3A-Context-Aware-Tiny-BERT-for-Detecting-Lee-Saxe/0940918e1bfe26bc4592e62da2f330474fad8420***](https://www.semanticscholar.org/paper/CATBERT%3A-Context-Aware-Tiny-BERT-for-Detecting-Lee-Saxe/0940918e1bfe26bc4592e62da2f330474fad8420)

***[9] –*** ***DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features. (2020, October 1). ScienceDirect.*** [***https://www.sciencedirect.com/science/article/abs/pii/S1568494620304440***](https://www.sciencedirect.com/science/article/abs/pii/S1568494620304440)

***[10] – Marchal, S. (2017). [PDF] Off-the-Hook: An Efficient and Usable Client-Side Phishing Prevention Application | Semantic Scholar. Semanticscholar.*** [***https://www.semanticscholar.org/paper/Off-the-Hook%3A-An-Efficient-and-Usable-Client-Side-Marchal-Armano/c13029692739c267cc68fdf679283aa94fb0f4bf***](https://www.semanticscholar.org/paper/Off-the-Hook%3A-An-Efficient-and-Usable-Client-Side-Marchal-Armano/c13029692739c267cc68fdf679283aa94fb0f4bf)

***[11] – Zhang, Y. (2007). [PDF] Cantina: a content-based approach to detecting phishing web sites | Semantic Scholar. Semanticscholar.*** [***https://www.semanticscholar.org/paper/Cantina%3A-a-content-based-approach-to-detecting-web-Zhang-Hong/414927a4a02727e1842d1c4f388af9d0a25ecf08***](https://www.semanticscholar.org/paper/Cantina%3A-a-content-based-approach-to-detecting-web-Zhang-Hong/414927a4a02727e1842d1c4f388af9d0a25ecf08)

***[12] – Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi (2012) An Assessment of Features Related to Phishing Websites using an Automated Technique. In: International Conferece For Internet Technology And Secured Transactions. ICITST 2012 . IEEE, London, UK, pp. 492-497. ISBN 978- 1-4673-5325-0 - This version is available at*** [***http://eprints.hud.ac.uk/id/eprint/16229/***](http://eprints.hud.ac.uk/id/eprint/16229/)

***[13] –Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi (2012) An Assessment of Features Related to Phishing Websites using an Automated Technique. In: International Conferece For Internet Technology And Secured Transactions. ICITST 2012 . IEEE, London, UK, pp. 492-497. ISBN 978-1-4673-5325-0***

***[14] –* *Mohammad, Rami, Thabtah, Fadi Abdeljaber and McCluskey, T.L. (2014) Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25 (2). pp. 443-458. ISSN 0941-0643***

***[15] – Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi Abdeljaber (2014) Intelligent Rule based Phishing Websites Classification. IET Information Security, 8 (3). pp. 153-160. ISSN 1751-8709***

***[16] – Detection of malicious Encrypted Web Traffic Using Machine Learning by Jay Shah B.Eng. Gujarat Technological University, 2014*** [***https://dspace.library.uvic.ca/bitstream/handle/1828/10313/Shah\_Jay\_MEng\_2018.pdf?sequence=3&isAllowed=y***](https://dspace.library.uvic.ca/bitstream/handle/1828/10313/Shah_Jay_MEng_2018.pdf?sequence=3&isAllowed=y)

***[17] –*** ***A. (2020). GitHub - arunabellgutteramesh/benign-phishing-url-classification-using-whois-and-lexical-features: To identify and extract features from URL that help classify URLs into benign/phishing and train an ML model with these features for classification. GitHub.*** [***https://github.com/arunabellgutteramesh/benign-phishing-url-classification-using-whois-and-lexical-feature***](https://github.com/arunabellgutteramesh/benign-phishing-url-classification-using-whois-and-lexical-feature)

***[18] – CIC-Bell-DNS 2021 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. (2021). Www.Unb.ca/Cic.** <https://www.unb.ca/cic/datasets/dns-2021.html>

***[19] – Classifying Malicious Domains using DNS Traffic Analysis*. (2021, October 1). IEEE Conference Publication** | IEEE Xplore. <https://ieeexplore.ieee.org/document/9730600>

***[20] –* Alaidaros, H. (2018, August 4). *An Overview of Flow-Based and Packet-Based Intrusion Detection Performance in High Speed Networks***. **Www.Academia.Edu.** <https://www.academia.edu/37181904/An_Overview_of_Flow_Based_and_Packet_Based_Intrusion_Detection_Performance_in_High_Speed_Networks>

***[21] -* Sharma, A. (2020, May 12). *Decision Tree vs. Random Forest - Which Algorithm Should you Use?* Analytics Vidhya**. <https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/>

***[22] -* Saini, A. (2021, August 26). *Logistic Regression | What is Logistic Regression and Why do we need it?* Analytics Vidhya.** <https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/>

***[23] -* Hachcham, A. (2021, July 19). *The KNN Algorithm - Explanation, Opportunities, Limitations*. Neptune.Ai.** <https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations>

***[24] –* Bento, C. (2022, January 5). *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis*. Medium.** <https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141#:%7E:text=A%20Multilayer%20Perceptron%20has%20input,use%20any%20arbitrary%20activation%20function>.

***[25] -* Majumder, P. (2020, February 23). *Gaussian Naive Bayes*. OpenGenus IQ: Computing Expertise & Legacy**. <https://iq.opengenus.org/gaussian-naive-bayes/>

***[26] -* HIDAWI, K. (2022, August 1). *Network Packet level –based Intelligent Phishing Intrusion Detection System (NIPDS)*. Github. Retrieved August 1, 2022, from** <https://Khaoulahidaawi.github.io/NIPDS/>

 ******



***Realized By******Khaoula Hidawi***

***Under supervision of******Dr. Ismail Berrada***

***INSTITUTE OF SCIENCE, TECHNOLOGY***

***AND INNOVATION (IST&I)***

***UM6P-CS***