EXPLORING PHISHING DETECTION USING SEARCH ENGINE OPTIMIZATION AND UNIFORM RESOURCE LOCATOR BASED INFORMATION

by

Kewei Ma

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2021

I would like to dedicate this thesis to my parents.

Table of Contents

List of Tables . . . . . **v**

List of Figures . . . . . **vii**

Abstract . . . . . **ix**

List of Abbreviations and Symbols Used . . . . . **x**

Acknowledgements . . . . . **xii**

Chapter 1 Introduction **1**

Chapter 2 Literature Review **4**

2.1 Datasets 5

2.2 Summary 6

Chapter 3 Methodology **7**

3.1 Data Collection and Preparation 7

3.2 Feature Sets 11
3.2.1 UCI Feature Set 11
3.2.2 CANTINA+ Feature Set 13
3.2.3 Proposed Feature Set 16

3.3 Machine Learning Classifiers 19
3.3.1 Decision Tree 19
3.3.2 Random Forest 19
3.3.3 Support Vector Machine 20

3.4 TPOT Automated ML Tool 21

3.5 DOM Analysis 23

3.6 Evaluation Metrics Used 24
3.6.1 TPR and FPR 24
3.6.2 F1-score 25
3.6.3 AUC Value 26

3.7 Proposed Thesis Approach 26

3.8 Summary 28

Chapter 4 Results and Discussion 29

4.1 Evaluations 29

4.2 Training Results Using Various Classifiers 30

4.3 Phishing Links percentage in Training Dataset 33

4.4 Proposing a new feature set: Categories of Features 33

4.5 Finalizing the Proposed the Feature Set 35

4.6 Comparison Between Grid Search Decision Tree and TPOT Pipeline . 40

4.7 Web API for Users 41

4.8 Summary 42

Chapter 5 Conclusion and Future Work 48

Bibliography 51

Appendix A Figures 55

List of Tables

3.1	Sources for legitimate and phishing links	8
3.2	Details of the datasets used in the research	10
3.3	UCI Feature Set	13
3.4	CANTINA+ Feature Set	15
3.5	Complete list of Classifiers and Regressors offered by TPOT	21
4.1	Baseline evaluations using UCI Feature Set	30
4.2	Training/Validation Results with Dataset-1 and UCI features (without "F5 — Having IP Address" and "F9 — Web Traffic").	31
4.3	Training results on Dataset-1 with URL-based and HTML-based features from CANTINA+ feature set with different threshold values for "F9 — Non-matching URLs"	32
4.4	Training/Validation Results with Dataset-1 and CANTINA+ feature set.	32
4.5	Training F1-score for each feature category.	34
4.6	Training TPR and FPR for each feature category	34
4.7	Default List of Sensitive Words Actual Frequency in All Legitimate and Phishing URLs	38
4.8	Training results with all 17 proposed features using grid search Decision Tree	39
4.9	Samples of Combination of Features with URL-based only Training/Validation Results	40
4.10	Comparison between TPOT and Decision Tree classifier	41
A.1	Training/Validation Results with Dataset-1 and UCI features (without "F5 — Having IP Address" and "F9 — Web Traffic").	59
A.2	Training/Validation Results with Dataset-1 and CANTINA+ feature set.	59
A.3	Training F1-score for each feature category using multiple classifiers	60

A.4 Training TPR and FPR for each feature category using multiple classifiers 60

List of Figures

1.1 A screenshot of the information page on PhishTank [8]. 2

3.1 A screenshot of "https://www.google.com/" SEO provided statistic using Alexa Top Sites. 9

3.2 Sample of dataset, where "1" represents "phishing" and "0" represents "legitimate". 10

3.3 Syntax of an HTTP URL [18]. 11

3.4 A screenshot of "https://penguin-stats.io/" SEO tool statistics on Alexa Top Sites. 12

3.5 An Example Visualization of a Decision Tree on the Iris Dataset. 20

3.6 Example of the TPOT pipeline [23]. 22

3.7 Illustration of a Confusion Matrix. 25

3.8 An example ROC curve plotted using Logistic Regression with random data points. 27

3.9 Overview of the proposed approach. 28

4.1 Visualization for the Decision Tree generated with UCI features. 43

4.2 Area Under ROC Curve (AUC) bar plot for UCI feature set with Random Forest training results. 44

4.3 Area Under ROC Curve (AUC) bar plot for CANTINA+ feature set with Random Forest training results. 44

4.4 Phishing links percentages vs. Training F1-score and TPR. . . 45

4.5 File size in kB for the HTML documents of legitimate websites (left) and phishing websites (right). 45

4.6 Wordcloud for word phases in all phishing URLs. 46

4.7 Wordcloud for word phases in all phishing URLs. 46

4.8 A complete generation of the TPOT optimizer. 47

4.9 Screenshot when using the web API to predict the legitimacy of "https://www.dal.ca/". 47

5.1 An illustration of the Model Search Working Algorithm. [21]. . 50

A.1 Validation results' F1-score and TPR for Generation 1 with grid search Decision Tree 55

A.2 Validation results' F1-score and TPR for Generation 2 with grid search Decision Tree, with "Number of ?" removed. The removed feature has the highest validation F1-score and TPR when removing the feature from the set. 56

A.3 Validation results' F1-score and TPR for Generation 3 with grid search Decision Tree, with "Out-of-position TLD" removed . . 56

A.4 Validation results' F1-score and TPR for Generation 4 with grid search Decision Tree, with "Embedded Domain" removed . . . 57

A.5 Validation results' F1-score and TPR for Generation 5 with grid search Decision Tree, with "Subdomains" removed 57

A.6 Validation results' F1-score and TPR for Generation 6 with grid search Decision Tree, with "Number of =" removed. 58

A.7 Validation results' F1-score and TPR for Generation 7 with grid search Decision Tree, with "PageRank" removed. The process stopped as we reached feature number eight, both F1-score and TPR of the model dropped below 0.97, which is the threshold we set for this process. 58

Abstract

Phishing attacks are the work of social engineering. They are used to trick users to obtain their sensitive / private information using malicious links, web sites and electronic messages. In this thesis, phishing attack detection is explored using information based on uniform resource locators (URLs) and third-party search engine optimization (SEO) tools. A supervised learning approach is used to detect phishing web sites. Evaluations are performed using real world data and a Decision Tree model, which optimized using the Tree based Pipeline Optimization Tool (TPOT) via Automated Machine Learning (AutoML). The results obtained are not only better than the state-of-the-art models in the literature, but also achieve 97% detection rate. To utilize the proposed model, the best-performing pipeline from TPOT is embedded to a web API for future remote access.

List of Abbreviations and Symbols Used

AC	Associative Classification
APAC	Anti-Phishing Alliance of China
API	Application Programming Interface
APWG	Anti-Phishing Working Group
ARFF	Attribute-Relation File Format
ASCII	American Standard Code for Information Interexchange
AUC	Area Under ROC Curve
AutoML	Automated Machine Learning
CANTINA	Carnegie Mellon Anti-phishing and Network Analysis
DNS	Domain Name System
DOM	Document Object Model
FP	False Positive
FPR	False Positive Rate
GSB	Google Safe Browsing
HTTP	Hypertext Transfer Protocol
IDF	Inverse Document Frequency
IP	Internet Protocol
KNN	K-Nearest Neighbors
LSTM	Long Short Term Memory

ML	Machine Learning
QDA	Quadratic Discriminant Analysis
REST	REpresentational State Transfer
ROC	Receiver Operating Characteristic
SEO	Search Engine Optimization
SHA	Secure Hashing Algorithm
SMS	Short Message Service
SVM	Support Vector Machine
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TP	True Positive
TPOT	Tree based Pipeline Optimization Tool
TPR	True Positive Rate
UCI	University of California, Irvine
URL	Uniform Resource Locator
VoIP	Voice over Internet Protocol
WSGI	Web Server Gateway Interface

Acknowledgements

I would like to express my gratitude towards my research advisors, Dr. Nur Zincir-Heywood and Dr. Riyad Alshammari. Without their guidance and dedicated involvement in every step of my research, this thesis would have never been accomplished. I would also like to thank my family and friends for their company during the COVID-19 pandemic, who stayed close to me when I am depressed.

Chapter 1

Introduction

Phishing attacks are the work of social engineering. Such attacks cannot be prevented only by setting up firewalls, building network defense systems, or encryption mechanisms. Instead of targeting the computer, phishing attacks target users who used the networks and systems. Initially, phishing attacks are launched by sending out legitimate-looking emails to lure people into clicking the links maliciously created by the hackers. According to Verizon's 2018 Data Breach Investigations Report, email is the number one vector for both malware distribution (92.4%) and phishing (96%) [8][38]. Nowadays, hackers are no longer satisfied with simple emails, it has spread beyond to include voice over internet protocol (VoIP), short message service (SMS), instant messaging, social networking sites, and even massively multiplayer video games [17].

Regular phishing attacks often involve three phases: (i) potential victims receiving the phishing links that could direct them to the malicious websites; (ii) the victims visit the phishing websites, resulting in providing sensitive information to the hackers (without recognizing) or ending up with backdoors installed on their computers for remote access (without knowing); and (iii) hackers monetizing stolen information or launching remote attacks [17].

The purposes of such attacks are to trick people into sharing private information, such as bank card numbers, accounts, passwords, or even one's identity on the Internet. More offensive phishing attacks could lure the victims to download unwanted software and malicious contents without being aware of it, or the website itself could contain JavaScript miners to occupy the victim's computer's resources for cryptocurrency mining. The fake websites created by the hackers often look very similar to the popular brands and electronics market places [36]. For example Figure 1.1 shows a phishing site that imitates to be Microsoft's sign-in page.

Phishing detection mechanisms mainly fall into three categories: (i) Relying on

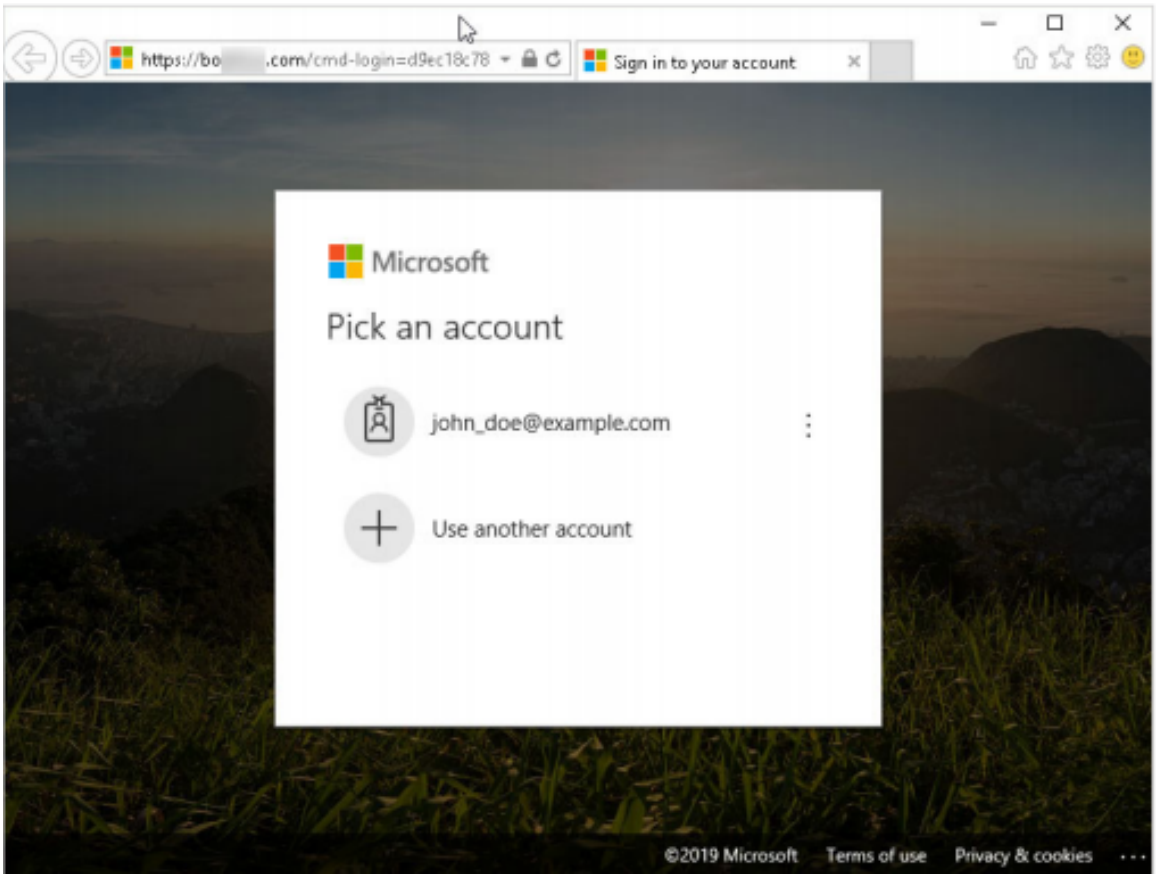

Figure 1.1: A screenshot of the information page on PhishTank [8].

humans' common sense; (ii) Requiring help from blocklists; or (iii) Employing machine learning algorithms. Phishing attacks are targeting users' curiosity and the sense of insecurity they felt when it comes to online banking or account passwords. Thus, educating the users about the existence of such attacks and how to protect themselves when encountering suspicious websites is essential. It does not matter if the victims (users) are part of Top 500 worldwide organizations or individuals, they could easily fall into such traps even when they may have the relevant training.

Blocklist-based approaches such as Google Safe Browsing, which requires an enormous size of a frequently updated online database to store all the reported phishing links, the number of resources behind such systems are often unthinkable. Systems supported by machine learning models are a popular research trend in recent years. However, such systems often require complicated feature extraction algorithms, slowing down the response time when processing individual links, making them impossible

to achieve a near real-time phishing attack (link) detection.

The objective of my research is to design and develop a light weight, near real-time machine learning model for phishing detection. To achieve this, I aim to explore uniform resource locator (URL) and search engine optimization tool based features for training the machine learning based phishing detector. In doing so, my aim is to be able to use such a trained model that requires minimum amount of feature extraction process to predict the legitimacy of an unknown URL. Eventually, the model will be open-sourced and for non-commercial users, with a simple web API for remote access. Thus, the research contributions for this thesis are as follows: (1) Designing and building a TPOT (Tree based Pipeline Optimization Tool) optimized decision tree-based model, which is implemented with the minimized amount of features selected in a best-performed feature set that could be extracted from any given URL without having to access the suspicious websites; and (2) The model can be easily integrated into a web API, allowing the users to query the legitimacy of a URL anytime and anywhere. In this work, the feature set that is explored is inspired by the pipeline constructed by Xiang et al. [39]. However, I extend on the features compared to Xiang et al.'s work and aim for a more effective feature extraction mechanism. With all that combined, evaluations show that my proposed system achieves a better prediction rate than average ML based detection models.

The remainder of this thesis is structured as follows. Chapter 2 reviews the recent research on phishing detection as well as compares the different mechanisms and approaches. Chapter 3 discusses the methodology followed in this thesis, including data collection, supervised learning algorithms, and the model's evaluation metrics. Chapter 4 presents the evaluation, results of my experiments and a description of the web API that I designed and developed. Finally, Chapter 5 concludes the research findings and discusses the possible future work.

Chapter 2

Literature Review

For machine learning-based anti-phishing systems, learning the features of a suspicious website is one of the most important aspects of phishing detection. Salihovic et al. have shown in their study that the machine learning (ML) approaches are the most effective methods to detect phishing, and at the same time, features in the ML models also contribute positively towards the prediction accuracy [32]. Phishing detection using machine learning can also be divided into two categories: (i) from the perspective of the programming level and the structure of a website; and (ii) from the perspective of the visual analysis of a web page.

For analysis of the website's structure and technical details, Bo et al. proposed a system to detect phishing activities based on domain name system (DNS) query logs and known phishing URLs [5]. They extracted the suspicious hosts from the DNS logs, combined with the Top N most frequent phishing URL's pathname, to generate a new phishing link. If the generated link exists and does not pass the false alarm filter, then the said link can be labeled as a phishing URL. The phishing database that was used in this study came from the Anti-Phishing Alliance of China (APAC), which is the authoritative anti-phishing organization whose main duty is collecting all phishing reports in China and doing appropriate handling on the real phishing attacks as quickly as possible.

Abdelhamid et al. introduced a new intelligent approach based on data mining called Associative Classification (AC) that could effectively detect phishing websites with high accuracy [1]. The authors used the data collection from the online phishing community PhishTank as their primary phishing source, as well as the data received from the Anti-Phishing Working Group (APWG) which maintains a "Phishing Archive" describing phishing attacks since 2006 [15]. The legitimate data source was not mentioned in the paper. Tan et al. leveraged the website hyperlink structure to create a web graph for C4.5 Decision Tree training [35]. However, they did not share

the exact sources of the data that was used in the paper.

Apart from the Decision Tree classification method, Support Vector Machine (SVM), Naïve Bayes and other ML models also show good performances when detecting phishing websites. The feature-based enhanced anti-phishing system proposed by Orunsole et al. used the SVM and Naïve Bayes to train their 15-dimensional feature set. The features that were used this research were extracted from the URL itself, the content of the web page, along with the website's behavior on the Internet. They reported a surprisingly high, 99% accuracy, for both classifiers [25]. However, if a malicious website chose to mimic the appearance of a legitimate site, or the site is heavily based on images and videos, the detection accuracy of this approach is shown to drop.

Moreover, a language-independent model called PhishDump that was built by Srinivasa et al. is based on the multi-model ensemble of Long Short Term Memory (LSTM) and SVM classifier, which could perform phishing detection on mobile devices [31]. The limitation of this technique is that when it encounters a new URL structure / pattern, or if the URL is using a shorten service ("bit.ly" or "ht.ly"), the performance drops and it could fail to detect the legitimacy of the link.

Some other researchers chose to analyze the contents of the web page, or they generated a visualization of the web page itself, analyze the similarity or compare the design layout between phishing websites. Adebowale et al. used the related features of images, frames, and text of websites for ML model training [2]. Patil et al. built a hybrid phishing detection pipeline that analyzed the features of the URL itself, the technical information behind the website (IP address, Registrar domain server, etc.), and also the visual appearance of the website [26]. Bozkir et al. proposed a model that could recognize the brand logos in a web page screenshot by solely using computer vision methods for object detection [6].

2.1 Datasets

To the best of my knowledge there are three online phishing website databases, which permits access via an API. These are: Google Safe Browsing (GSB), OpenPhish, and PhishTank. GSB has been available since 2007, it is to protect its users from web-based threats like malware, unwanted software, and social engineering across

desktop and mobile platforms [13]. To determine if a URL is on any blocklists of Safe Browsing, developers can send an HTTP POST request with an actual non-encrypted URL to the Google server, the server will then respond whether the URL is safe or not. Alternatively, developers can also download the current version of the database from Google. However, the local copy of the GSB blocklist is encrypted as variable-length SHA256 hashes, meaning we could not obtain the actual phishing URLs from those sources. This makes it impossible for developers to analyze the patterns of phishing URLs themselves. Moreover, given that this is intended for developers to use, most end users are not even aware of it.

On the other hand, PhishTank and OpenPhish both offer non-encrypted datasets of phishing URLs. The difference is that PhishTank is a free community site where anyone can submit, verify, track and share phishing data [28]. While OpenPhish is a fully automated self-contained platform for phishing intelligence [24]. From Bell et al. [4]'s experiments, during the one-month data they collected, OpenPhish's online database added 53,234 unique phishing URLs with 14,721 unique domains, while PhishTank added 48,473 unique URLs with 20,458 unique domains (3,761 fewer URLs compare to the OpenPhish database) over the same time period. Again, these are intended for developers to use, and most end users are not aware of them.

2.2 Summary

In this chapter, the two categories of feature-based phishing detection systems are presented. The first category of works in the filed are focused on the programming level, and the others are focused on visual analysis of the target website's design layout. Additionally, by analyzing the classification algorithms that were used in previous research, it is found that Decision Tree, Random Forest, and Support Vector Machine are the most commonly used in feature-based phishing detection systems. Also, publicly available phishing and legitimate datasets that were used in the aforementioned studies are introduced. For phishing web data, PhishTank and OpenPhish are the two publicly accessible online datasets that were used in multiple works. Most researches do not share the exact source of their legitimate data, Alexa's Top Site service is the only known publicly available source for obtaining legitimate URL data.

Chapter 3

Methodology

In this section, I will discuss my approach to obtain the best supervised learning (classification) model based on the proposed SEO and URL based features that are used to detect phishing websites. I will start with the data collection procedures and feature selection mechanisms, presenting the datasets and feature sets that are used at different stages of my research. Then, I will move on to the machine learning classifiers that are used in all the experiments in this study, including the automated machine learning tool, TPOT, which is introduced to improve and finalize my model, followed by the metrics that are used to evaluate the performance the classifiers. Finally, I will conclude this section with the detailed approach I took, along with a workflow diagram to demonstrate the stages of this research.

3.1 Data Collection and Preparation

The quality and quantity of the data used in a machine learning based system is critical. Thus, in this research, I aim to carefully select the source for phishing and legitimate websites. Table 3.1 shows the details of my data sources and the dates they are collected.

The UCI Dataset is downloaded from the UCI Machine Learning Repository [1], contains 1353 data instances from multiple sources: the 702 phishing websites came from the online community-based database PhishTank, the 543 legitimate websites are collected from Yahoo. There are also third categories of labels, the 103 suspicious websites were crawled using a browser's plug-in PHP script.

During my thesis research, I have also started the data collection process in June, 2020. PhishTank-Jun and PhishTank-Jan are directly downloaded from PhishTank in 2020 and 2021, respectively. For PhishTank-Jun, I downloaded the latest phishing database every day at the same hour for one month, after combining all those datasets and eliminating the duplicate entries, I obtained 31,945 unique phishing

Table 3.1: Sources for legitimate and phishing links

Name	Labels	Sources	Created Date
UCI Dataset	Phish/Legit	UCI Machine Learning Repository.	November 2016
PhishTank-Jun	Phish	PhishTank online dataset.	June 2020
Top Sites-Jun	Legit	Alexa Top Sites Rank 1000-2000, crawled to depth two.	June 2020
PhishTank-Jan	Phish	PhishTank online dataset.	January 2021
Legit-Jan	Legit	Official websites including banking, education, government, social media; COVID-19 informational sites, etc., crawled to depth two.	January 2021

URLs. Among those URLs, I successfully downloaded 9,290 web pages, the others were either not responding or returning 400/404/503 HTTP errors. In this case, they might be permanently taken down from the Internet, or cannot be accessed using my Python scripts. According to Bell et al. [4], the average duration for phishing URLs to stay in the PhishTank blocklist is approximately 2 days, as they would be removed once they do not exist on the Internet. PhishTank-Jan is processed under similar conditions but was only downloaded once near the end of January 2021. It is used only as a test set to observe if the time difference (June 2020 vs January 2021) in the data collected would affect the model prediction's rate.

Top Sites-Jun is a set of legitimate data collected via Amazon Alexa's Top Sites API in June 2020. The Top Sites service is offered by Alexa, contains a list of websites ordered by their one-month Alexa traffic rank. The API also has options for downloading a certain range of websites or access the website list from a certain country or category.

Furthermore, many studies on analyzing the influence of a website rely on SEO tools such Alexa and SimilarWeb, however, neither of them can accurately reflect the actual traffic of a website [29]. A screenshot of the SEO statistic of Alexa Top Site is shown in Figure 3.1.

The Alexa Top Sites may not provide the most accurate site rank, but it is a

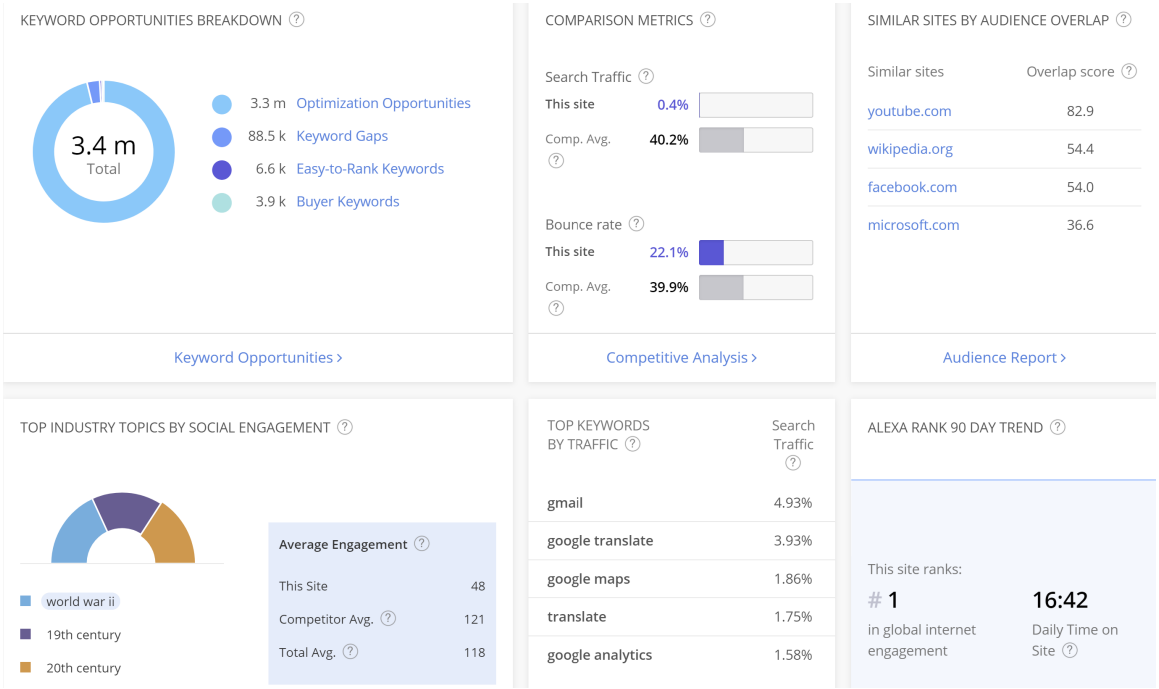

Figure 3.1: A screenshot of "https://www.google.com/" SEO provided statistic using Alexa Top Sites.

good service to look for legitimate websites. Therefore, I requested the URLs for all the web pages with sites ranked 1,000 to 2,000 in June, to include more legitimate websites in my data collections. I used a Python script to crawl all those 1,000 links I obtained.

URLs included in the data collection called Legit-Jan contains websites from education, government, social media, banking fields, such as many universities' official sites, Canadian government official site, Reddit's discussion threads, etc. The legitimacy, in this case, is mostly based on common sense given Alexa list of URLs.

The next step is to compile datasets that are suitable for evaluating the ML models at each stage. Table 3.2 shows the details of the datasets I built for this research. Apart from the UCI Dataset, all the other datasets are collected / crawled by me. Figure 3.2 shows an example of the dataset that was used in this research. Dataset-0 and Phish-1 consist of the data I collected in June 2020. The phishing and legitimate data instances in Dataset-1 are balanced, I trained and validated on this dataset to set the baseline for this research. However, after considering that this ratio might affect the model performance, I applied a series of experiments to observe the model's

Table 3.2: Details of the datasets used in the research

Dataset Name	Total Data	Legitimate Data	Phishing Data
UCI Dataset	1353	651	702*
Dataset-0	2464	1232	1232
Dataset-1	9000	1000	8000
Dataset-2	2000	1000	1000
Phish-1	9411	0	9411
Phish-2	10115	0	10115

*Suspicious URLs and Phishing URLs both considered as phishing in this case.

training result with various ratios. The detailed training procedures can be found in the results section. The preliminary empirical results for the phish/legit ratio was 8:1. This is reflected in Dataset-1. To sample the phishing data instances from Phish-1, I used the build-in method sample() from the Python Pandas library.

The data in Dataset-2 and Phish-2 were collected in January 2021. The purpose of this dataset is to test if the time difference in data collection dates might affect the model performance. Phish-2 represents all the phishing URLs I downloaded from PhishTank in one single month.

id	url	result
1399	https://yandex.com/?lang=id&sk=y6b0d012bd8aa31c5d33c7ec23727b20a	0
1400	https://youtu.be/t12x7k_oecc	0
1401	https://youtu.be/wKTfCVlKPYE	0
1402	https://youtube.googleblog.com	0
1403	https://yuer.pcbaby.com.cn/483/4834996.html	0
1404	https://zhuanti.lagou.com/TOP2020.html	0
1405	https://zoom.us/download#client_4meeting	0
1123978	http://creditiperhabbogratissicuro100.blogspot.com/2011/02/habbo-crediti-gra	1
1585505	http://www.jjscdc.cn/adfile/login.html?http://us.battle.net/login/en/?ref=http:/	1
1997235	http://bullwinsconfecciones.com/images/44.html	1
2042606	http://www.formbuddy.com/cgi-bin/formdisp.pl?u=Twice&f=LLM	1
2080508	http://alshurayet.jeun.fr	1
2637207	http://www.latos.co.kr/js/erusr/united/ep/meniu1.htm?theinfored=2718ad761	1

Figure 3.2: Sample of dataset, where “1” represents “phishing” and “0” represents “legitimate”.

3.2 Feature Sets

The features that are used to train my machine learning model can be categorized into three sets: (i) The UCI feature set; (ii) The CANTINA+ feature set; and (iii) My proposed feature set. The first two sets represent the two different state-of-the-art approaches from the literature of Abdelhamid el al.[1] and Xiang et al. [39], respectively. The models trained on those two sets provide a reference baseline for my research, allowing me to evaluate the ML models with my proposed feature set.

3.2.1 UCI Feature Set

The UCI dataset contains eight features, as shown in Table 3.3. When setting the reference baseline for my research, I remove Feature "F5 — Having IP Address" and "F9 — Web Traffic", and train the model with the remaining eight features. The reason I remove these features is that they are not available for other publicly available data that I collect and crawl. It should also be noted here that the terms I used to describe the different components of a URL are based on the syntax of a URL as shown in Figure 3.3.

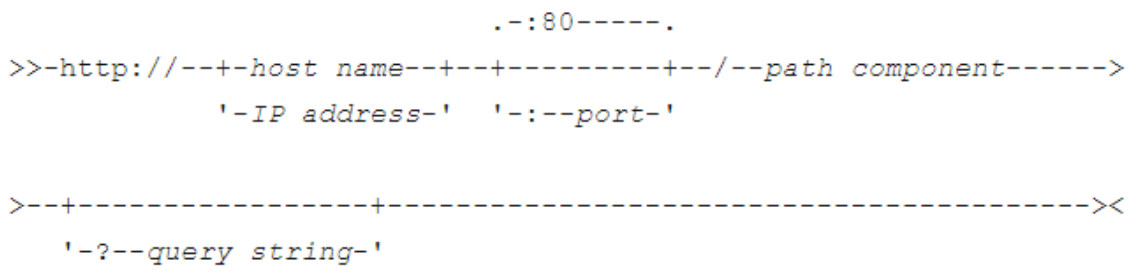

Figure 3.3: Syntax of an HTTP URL [18].

As mentioned before, the phishing websites used in this research are collected from PhishTank's online dataset, where the suspicious URLs are submitted by developers. In this case, I could not verify if the provided phishing IP addresses are correct. Moreover, even if I could obtain the IP addresses from other trustworthy third-party resources, the sites could easily hide their true IP addresses using a virtual private network or a proxy server. So, I opted not to use the IP address as a feature.

I also removed the "F9 — Web Traffic" from my training model. As discussed in Section 3.4, Alexa cannot provide accurate web traffic data and page rank for

websites. Also, Alexa can not generate web traffic data for phishing websites and other not so popular websites. For example, "https://penguin-stats.io/" is an online item inventory statistics tool created by a group of game players, the website is guaranteed as normal website, but Alexa web traffic tool can only generate keywords related to this website but no traffic data available (Figure 3.4). In this case, the uncertainty of those web traffic tools would misguide the machine learning models, especially for classifiers that heavily rely on the accuracy of data points. Thus, I decided to remove this feature from my feature set as well.

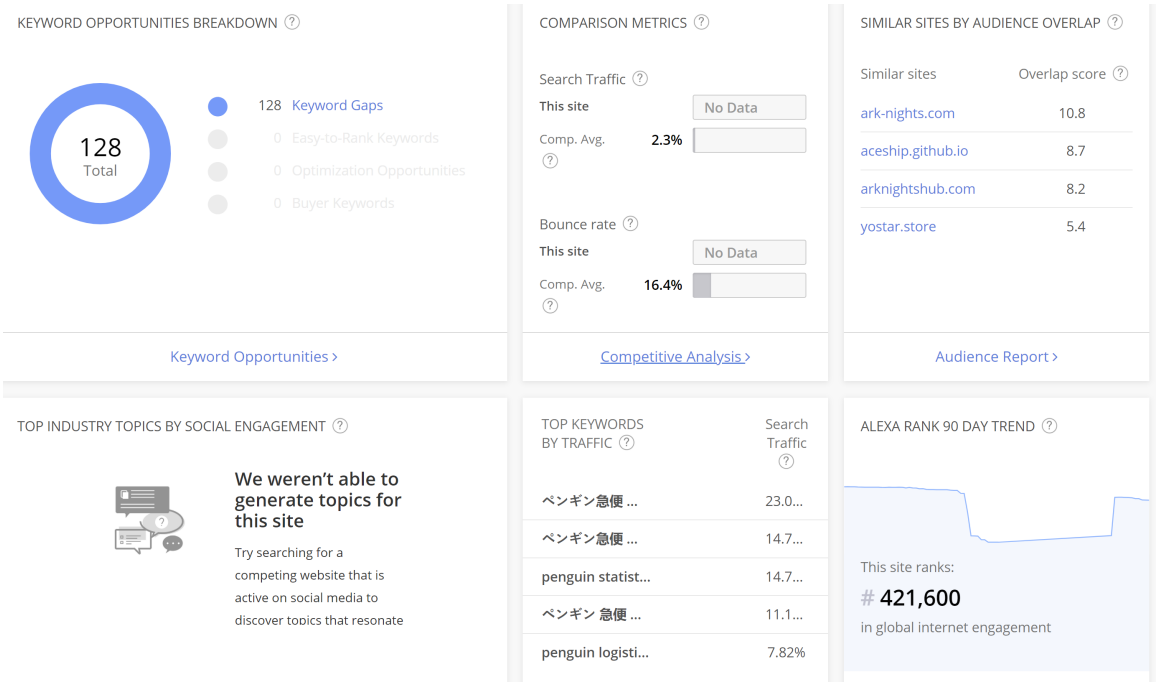

Figure 3.4: A screenshot of "https://penguin-stats.io/" SEO tool statistics on Alexa Top Sites.

Data provided by the UCI repository are encoded as Attribute-Relation File Format (ARFF). This is an ASCII text file that describes a list of instances sharing a set of attributes, it was developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software [27]. Thus, I set my encoding algorithms when processing the data I collected. For features F1, F2 F3, F6, and F7, if the target pattern is present in the URL or web page, I label them as True, else the value for the feature is set to False. Given that I would like to preserve as much information about the URL as possible,

the values of “F4 — URL Length” and ”F10 — Domain Age” features are stored as the actual numerical values of the URL length and the domain age. These are obtained from the domain registration date to February 2, 2021 in days. Thus, after feature extraction, I would need to normalize the data for the ML training process.

Table 3.3: UCI Feature Set

No.	Feature Name	Description
F1	UCI Anchor	If the anchor tag has “href” attribute
F2	Request URL	If the anchor tag contains URL
F3	Server Form Handler	Whether the form tag set “method” to “PUT” or “GET”
F4	URL Length	The length of the URL
F5	Having IP Address	If the websites has IP address
F6	Prefix/Suffix	If URL’s hostname contains “-” or “_”
F7	Domain is IP Address	If the URL’s hostname is an IP address
F8	Subdomain	If the URL has subdomains
F9	Web Traffic	The traffic of the website
F10	Domain Age	Date since the domain is registered

3.2.2 CANTINA+ Feature Set

Carnegie Mellon Anti-phishing and Network Analysis Tool (CANTINA), is a model that was proposed by Xiang et al. [39]. It is one of the most complex machine learning models in the literature. Thus, as I continued to analyze their feature set, I notice that some of the features could be removed based on their contribution to their overall performance. The CANTINA+ feature set groups the 15 features they employ into three categories: (i) URL-based; (ii) HTML-based; and (iii) Web-based. As the name

of the categories suggest, those features can be extracted from the URL, or from the HTML source of the web page, or can be obtained by combining existing information with online resources. Table 3.4 lists the details of the CANTINA+ features. In this thesis, this feature set will be referred to as the CANTINA+ feature set.

The technical details of how to extract the value for each feature are described in Xiang et al.'s paper [39]. It should be noted here that the CANTINA+ feature set include some features in common with the UCI feature set. To this end, the "F2 — IP Address" is the same as the UCI's feature "F7 — Domain is IP Address". Additionally, "F4 — Suspicious URL" is set if the domain name contains a dash("-") or the URL contains an "at" symbol("@"), which are the revised version of the UCI's feature "F6 — Prefix/Suffix". Moreover, "F11 — Age of Domain" is the same as the UCI feature "F10 — Domain Age".

As I analyze the CANTINA+ feature set, I also recognize that some of the features do not produce positive effects towards the training results. As one of the phishing URL examples provided by the authors of CANTINA+ paper is as the following:

"http://cgi.ebay.com.ebaymotors.732issapidll.qqmotorsqq.ebmdata.com"

On the other hand, the most frequent patterns in my collected dataset is as the following example:

"https://www.oncecall.in/css/imagecache/vgf.32454"

As it can been seen from these examples, the path component of the latter URL is significantly longer and more complicated that the first one from Xiang et al.'s dataset. Furthermore, I discover other issues when applying the CANTINA+ feature set to the data I collected for this research. This in return created the motivation to improve and extend the features based on the patterns observed in the new data (2020 / 2021) crawled and collected in my research.

- F4 — Suspicious URL: After processing the data, it was seen that even legitimate links could have "@" symbol in the URL, and the dash "-" is also very common in both legitimate and phishing URLs. Thus, this feature is extended as "F4 — Special Symbols" to avoid prejudice / bias when viewing the links.

- F7 — Bad Form: According to Xiang et al. [39], a URL must satisfy all of the following conditions to set this feature to True:

Table 3.4: CANTINA+ Feature Set

No.	Feature Name	Description
	URL-based Features	
F1	Embedded domain	If the path of URL contains domain format string
F2	IP Address	If the hostname is an IP address
F3	Number of Dots	Number of the dots in the whole URL
F4	Suspicious URL	If the URL contains "@" or the hostname contains "-"
F5	Number of Sensitive Words	The number of words in the word list* appear in the URL
F6	Out-of-position Top-Level Domain (TLD)	If the TLD appears anywhere abnormal
	HTML-based Features	
F7	Bad Form	If the page contains potentially harmful HTML form
F8	Bad Action Fields	If the action field contains suspicious contents
F9	Non-matching URLs	If all the URLs embedded in the HTML file has similar patterns
F10	Out-of-position Brand Name	If the most frequent domain name in all embedded URLs matches the target domain name
	Web-based Features	
F11	Age of Domain	Date since domain is registered
F12	Page in Top Search Results	Search the domain name with the top 5 most frequent words extracted by TF-IDF in Google
F13	PageRank	Result of Google Page Rank service
F14	Page in Top Search When Searching for Copyright Company Name and Domain	Search the Copyright name with domain name in Google
F15	Page in Top Search When Searching for Copyright Company Name and Hostname	Search the Copyright name with hostname in Google

  - Contains HTML form tag.
  - Contains HTML input tag as a child tag to form.
  - In the form tag, contains sensitive keywords such as “password”, “credit card”, or has no text message but only link points to a image.
  - A non-HTTPS scheme URL in the action field of the form tag, or the action field is empty.

  After processing the data used in this thesis, it was seen that no URL could satisfy all of the above conditions. Thus, this feature become irrelevant for the data used in this thesis.

- F9 — Non-matching URLs: This feature checks the patterns in all links embedded in the DOM object, if the occurrence of any patterns exceeds a given threshold, or the percentage of the empty anchor tag is over another given threshold, it is set as True. The thresholds are set as by Xiang et al. [39], and can vary given a dataset. Thus, in this thesis, the values of 30% and 50% of all the links are used to empirically determine the thresholds for both conditions. These results are given in the following chapter.

3.2.3 Proposed Feature Set

In this thesis, my goal is to explore and analyze different feature sets in order to determine and extract publicly available and easy-to-use features. To this end, after comprehensive analysis, 11 features are extracted. For this purpose, the URL itself and third-party SEO tools are used and different ML classifiers are trained to find the best performing model. The proposed features that are used in the model can be divided into two categories:

- URL-based features
- Web-based features

The URL-based features are extracted by only considering the URL itself. Characteristics such as the HTML tags count of the web page do not belong to this category.

- *F1 — IP Address* (Boolean): This feature is inherited from the CANTINA+ and UCI feature sets since it is used in both. It checks if the hostname of a URL is an IP address. This feature returns a Boolean value of whether the domain is an IP address or not. By analyzing the datasets UCI, Dataset-0, 1, and 2, it can be seen that if the domain of a website is an IP address, the possibility of it being a phishing website is extremely high.

- *F2 — URL length* (Numerical): This feature is inherited from the UCI and CANTINA+ feature sets as well [1] and [39]. This feature returns a numerical value of the length of the URL. Systems in the literature assume that the longer the URL is, the more likely it belongs to a phishing website. This assumption is considered as a possibility when collecting the data. However there are many legitimate websites with long URLs, too. As the authentication techniques improve, to ensure the security and privacy of the users, most auto-generated login confirmation links have longer lengths. Thus, those are also included in the datasets used in this research.

- *F3 — Shorten URL* (Boolean): URL shortening service is an online tool that could generate a shorter version of a long URL with the domain name of the shortening service and a random string with characters and numbers. The generated URL is often short and does not provide any information about the website. If the domain information for this kind of URL is checked, only the details of the domain that offers the shortening service will be returned. It would be impossible to determine the legitimacy of a short URL without having access to the website. This feature returns a Boolean value to reflect whether the link uses a URL shortening service or not.

- *F4 — Sensitive Words* (Numerical): This feature is developed from the "Number of Sensitive Words" feature in the CANTINA+ feature set and it has been adopted to fit our dataset better. The list that I used in this research is "secure", "account", "login", "signin", "confirm", "amp", "web". The procedures I took to generate this list can be found in Section 4.5. This feature returns the number of sensitive that appears in the URL.

- *F5 — Special Symbols* (Numerical): Phishing links tend to have symbols inserted in the URLs. The list that I used in this research to determine if the link contains special symbols is "-", "_", "@", "%". This feature extracts the total number of such special symbols present in the URL.

- *F6 — Number of Dots* (Numerical): This feature is similar to the feature "Subdomains", which counts the number of dot-separated string segments in the URL's netloc minus two to obtain the subdomain numbers. Thus, those two features are combined into one. This feature returns the number of dots present in the URL.

The web-based category contains features that can be determined with third-party SEO tools such as Google search engine, Whois domain lookup information etc. Note that, even though the Google PageRank contributes to the accurate prediction rate, the service has not been used by Google since 2006, according to the former Google engineer, Jonathan Tang [37]. Admittedly, there are many online tools for PageRank look-up.

- *F7 — Search Result* (Boolean): For branding purposes, legitimate websites tend to have a higher rank, making it easier for people to search for them. They are also more likely to use a unique and meaningful name as their domain name. Thus, if I would like to use Google search to check an unknown website's domain name, legitimate websites are more likely to appear in the first several results. This feature has a similar effect as the Page Rank feature, where Page Rank provides a rank that a web page has when a user search for a related keyword, so I combine these two features into one. This feature returns the result if searching for the URL's domain name in Google, the returned top N results contains the targeted doamin.

- *F8 — Domain Age* (Numerical): This feature extracts the days since the URL's domain is registered.

3.3 Machine Learning Classifiers

In this thesis, multiple machine learning classifiers are evaluated, including Logistic Regression, Decision Tree, Random Forest, KNN, SVM, Gaussian Process, with the primary goal of evaluating the effectiveness of the proposed feature set against the UCI and CANTINA+ feature sets on the same data. After extensive experiments, the Decision Tree classifier is chosen due to its high performance as well as human-understandable model solution. In this section, I will be describing the Decision Tree classifier and steps I took to obtain the accurate model training results, along with a brief introduction to the Random Forest and Support Vector Machine classifiers, as they are the most frequently used classifiers in similar problems.

3.3.1 Decision Tree

In very simplistic terms, a Decision Tree model is a flowchart-like structure that asks a question on every node based on one attribute of the data, split the records into two groups based on their values. Each internal node in the tree represents a test on an attribute, while the leaf denotes the results of the said test. Decision trees can also handle data with ten to hundreds of features[33]. As a transparent solution model, the training results of a Decision Tree can be interpreted via tree visualization tools such as Graphviz. To minimize the issue of overfitting, the cost_complexity_pruning technique provided by the scikit learn library is used to prune the Decision Tree in the valuations performed in this thesis. Figure 3.5 illustrates a sample decision tree generated using the Iris Dataset.

3.3.2 Random Forest

Random Forest is an ensemble of decision trees that can handle high-dimensional classification and regression problems. One of the ensemble methods is bagging, also known as bootstrap aggregation, and boosting, proposed by Breiman [7] in 1996. In this method, a random sample of data in the training set is selected to a subset. The model then train on each subset of data independently, which will ensure low correlation between the trees, and can also overcome the overfitting issue that is common in general decision trees. The result of the classifier is voted by the majority

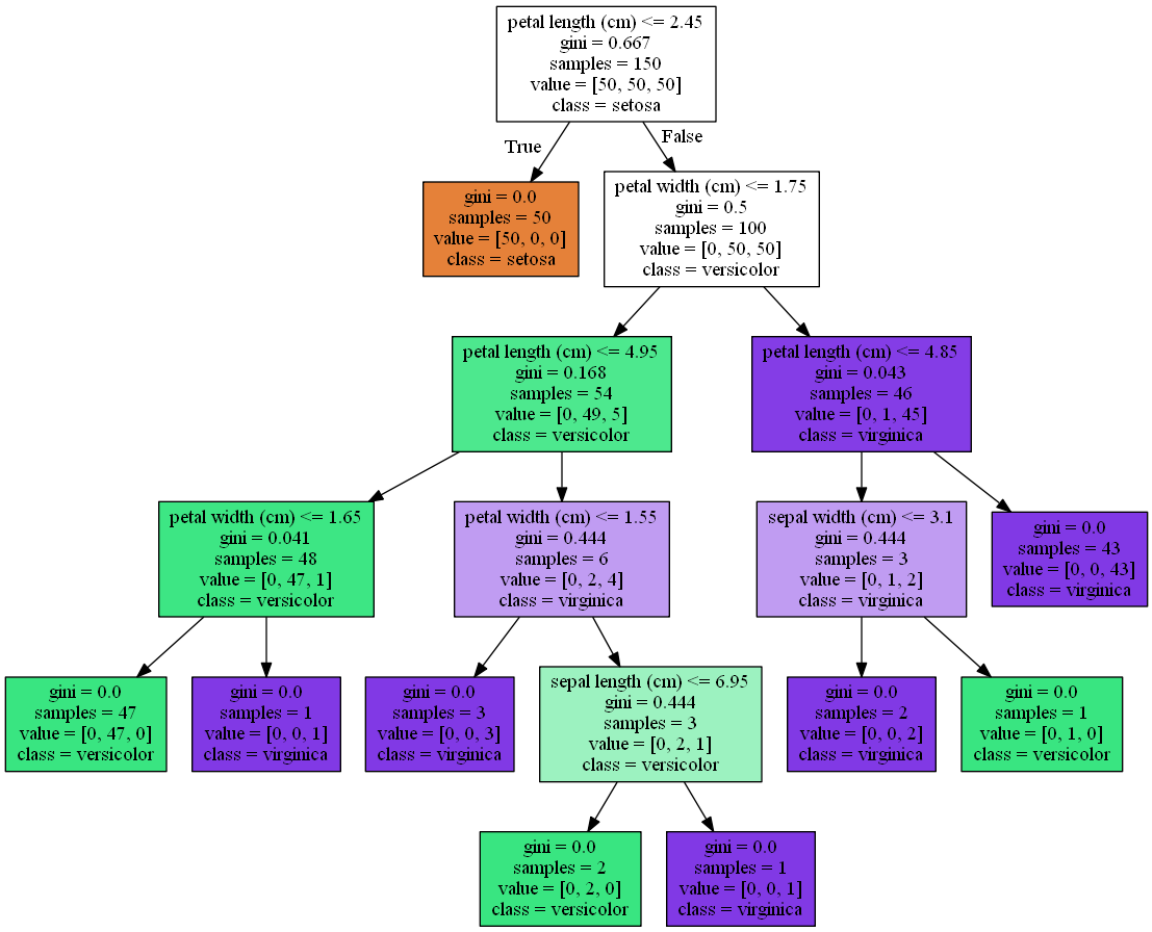

Figure 3.5: An Example Visualization of a Decision Tree on the Iris Dataset.

output of the trees, giving a more accurate prediction. It computes the best split in each node in terms of impurity reduction. However, the impurity computations often result in high computational cost during the training process [11].

3.3.3 Support Vector Machine

Support Vector Machine (SVM) classifier is a binary classification algorithm that is shown to be effective for discovering informative features or attributes in a high dimensional classification problem. Detecting the website's legitimacy is a binary problem, making this classification algorithm suitable for this research. SVM training identifies a small subset of the training data relevant for the construction of the classification boundary which forms the support vectors [30].

3.4 TPOT Automated ML Tool

Automated Machine Learning refers to the technique that could automatically discover well-performed models for supervised learning tasks with little to no human involvement. In this research, TPOT as an extention to AutoML toolset is used to generate an optimized machine learning pipeline for phishing detection. In general, TPOT initializes a genetic programming algorithm iterating through different preprocessing approaches, machine learning models, and their respective parameters, with an evolutionary algorithm to find the most suitable pipeline that could score high accuracy [12]. The default TPOT configuration that was used in this research has two operations: (i) Classification which is designed for supervised classification tasks, and (ii) Regression which performs automated deep learning for supervised regression tasks. A list of offered classifiers and regressors can be found in Table 3.5.

Table 3.5: Complete list of Classifiers and Regressors offered by TPOT

Classifiers	**Regressors**
Gaussian NB	Elastic Net CV
Bernoulli NB	Extra Trees Regressor
Multinomial NB	Gradient Boosting Regressor
Decision Tree Classifier	Ada Boost Regressor
Extra Trees Classifier	Decision Tree Regressor
Random Forest Classifier	K Neighbors Regressor
Gradient Boosting Classifier	Lasso Lars CV
K Neighbors Classifier	Linear SVR
Linear SVC	Random Forest Regressor
Logistic Regression	Ridge CV
XGB Classifier	XGB Regressor
SGD Classifier	SGD Regressor
MLP Classifier	

TPOT uses a recursive tree-based approach, and in each generation, every pipeline receives a copy of the dataset, as the node of the tree makes a prediction, the result is then saved as a new feature. An example of the pipeline is illustrated in Figure 3.6.

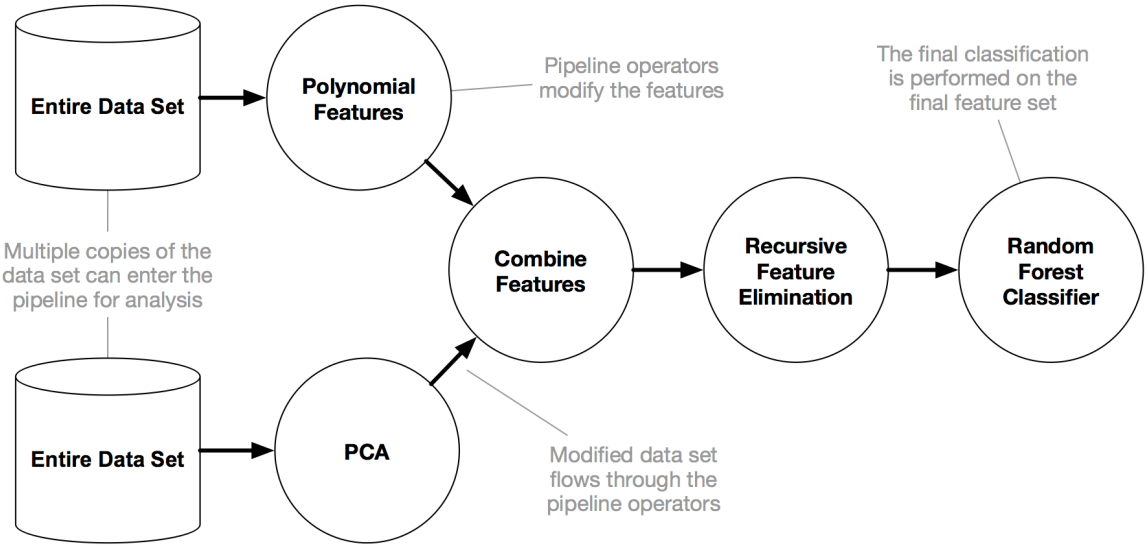

Figure 3.6: Example of the TPOT pipeline [23].

Moreover, there are other automated machine learning tools that could also perform the optimization process, including autoWeka, auto_ml, H2OAutoML, and auto-sklearn [20].

Auto_ML is a framework designed for a company's production system, which allows them to quickly extract values from data on their customer. It provides automation on the processing of the data including the feature engineering process through TF-IDF processing, data process, categorical encoding, and numeric feature scaling; machine learning processes including model construction, tuning, selection, and ensembling process [3].

Auto-sklearn is built around the scikit-learn library, and it focuses on small to medium-sized datasets. The main advantage of this system is that it automatically takes the past performance on similar datasets into account, saving the processing time by constructing ensembles from the models evaluated during the optimization. Similar to autoWeka, Auto-sklearn also uses Bayesian search for the best-performed model but added a racing mechanism for quick access [10].

The framework of H2O is similar to the sklean library. It is a collection of machine learning algorithms that can be executed on a variety of interfaces and can be accessed by multiple programming languages. The H2O autoML uses the algorithms provided in H2O to build the pipeline, the supervised algorithms support classification and regression problems [16].

A benchmarking on automatic machine learning tools, including auto-sklearn, auto_ml, H2O AutoML, and TPOT, are done by Balaji et al.. To ensure an accurate and fair statement, they tested the tools on a selection of 87 open datasets from OpenML and an online repository of standard machine learning datasets consistently exposed through a REST API. Each tool was tested on both regression and classification datasets. According to their results, auto-sklearn performs the best on classification datasets, while TPOT performs the best on regression datasets [3]. Moreover, since all the ML classifiers and evaluation metrics used in this thesis are based on the scikit learn library, I would not want to introduce any complications by using classifiers from different sources. Thus, I opted for TPOT with H2O to optimize the ML pipeline.

However, it should be noted here that some research claims that the TPOT optimizer do not scale well [12]. When analyzing the best-performing pipeline for a large dataset, it would take a very long time to produce the final results, as most of these tools are essentially using brute force to try out different combinations of models and their regarding parameters. The datasets used in this thesis have no size greater than 10,000 entries, and the maximum processing time for TPOT pipeline to finish the processing is around five minutes. Thus, the aforementioned problem is not experienced in this thesis. However, bigger datasets may require a different automated ML tool, which could be investigated in the future work.

3.5 DOM Analysis

Many phishing attacks would try to trick people into sharing their information through a fake login form. The analysis on the form tag can be found in both the UCI [1] feature set and the CANTINA+ [39] features set, which are two main references that were used in this research. It is observed that the patterns for the "method" attribute in a form tag, which specifies the HTTP method for submitting the form data. Thus, a feature can be extracted by categorizing whether a DOM object uses only the "GET", "POST", or both methods. The input tag, which is inside the form tag is used to define a user input field, phishing web pages tend to leave this field empty, or insert keywords such as "account", "password" [39].

Text analysis is another aspect of DOM analysis, one of the most common text

evaluations is word frequency. In this research, the Term frequency-inverse document frequency (TF-IDF) is used to summarize the topic and area of interest for each website. TF-IDF is a numerical statistical approach that can reflect how important a word is to a document in a collection or corpus [34]. The Term Frequency (TF) of a word directly reflects the occurrence of it in the whole document, while the Inverse Document Frequency (IDF) measures how much information the word can provide by taking the logarithm of the number of documents that contain the word divided by the total number of documents. TF-IDF then represents the results of TF and IDF to obtain the actual weight of the target word in all documents. Therefore, stopwords such as "at", "the" will not receive higher weight even though they might appear in every document. After eliminating the stopwords and filtering out documents that have file sizes smaller than a threshold, the top five keywords that weighted the most in a DOM object are obtained. Combining this information with the domain name of a web page, a search engine is used to look up the target website. The intuition behind this is that the legitimate website tends to have a higher rank in the search results when searching within a domain in the related area, while phishing websites mostly cannot reach a high rank in search results.

3.6 Evaluation Metrics Used

To evaluate the performance of each model, the True Positive Rate (TPR), False Positive Rate (FPR) and F1-score are calculated using the confusion matrix. TPR and FPR can reflect if the model correctly predicts the phishing links, F1 on the other hand, gives us a balanced overview of the model's performance. Since in this research one of the goals is to explore the best practice feature set, the effect of each feature on the performance needs to be understood, this is where the Area Under ROC Curve (AUC) comes in.

3.6.1 TPR and FPR

A general confusion matrix is similar to Figure 3.7, where the "Positive" represents the "Phishing" label, "Negative" represents the "Legitimate" label (Figure 3.7). In the follwoing experiments, the focus is on the True Positive (TP) and False Positive (FP) rates. As a phishing websites detection model, a high TPR means successful

detection of an actual phishing site, a low FPR means avoiding mislabeling valid URLs as malicious. The TPR and FPR can be calculated using the equations below:

$$TPR = \frac{TruePositive}{AllPositives}$$

$$FPR = \frac{FalsePositive}{AllNegatives}$$

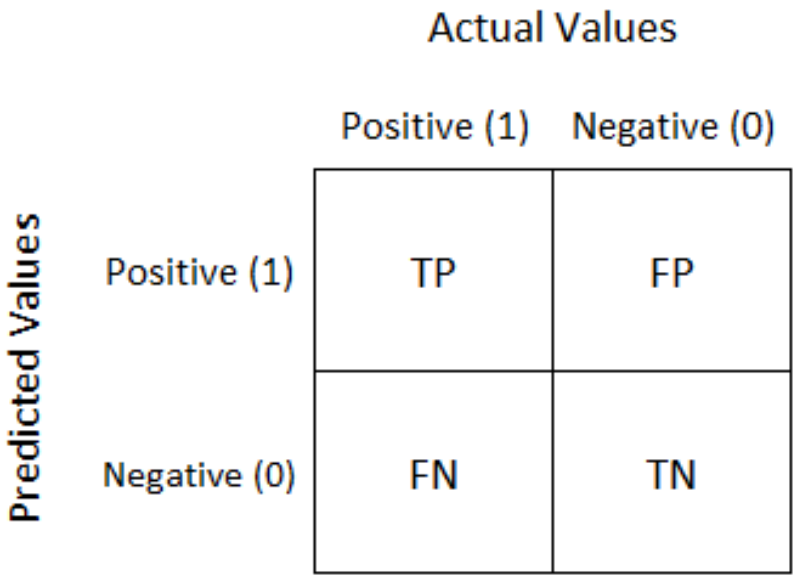

Figure 3.7: Illustration of a Confusion Matrix.

3.6.2 F1-score

The F1-score is the harmonic balance between precision and recall. Precision calculates the ratio between correctly classified (TP) data against all positive classes. If the result is high, then it means the model can identify most of the phishing websites. Precision is calculated using the following equation:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{TruePositive}{PredictedPositive}$$

Recall, on the other hand, calculates the ratio of the correctly classified (TP) data instances against the total number of the groundtruth of that class. The equation is

given below:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{TruePositive}{ActualPositive}$$

F1-score reports the weighted average of the precision and recall. In this thesis, F1-score function provided by the scikit-learn library (version 0.24.1) is used for model selection. F1-score can be calculated using the equation below:

$$F_\beta = (1 + \beta^2)\frac{precision \times recall}{(\beta^2 \times precision) + recall}, \quad where \beta = 1$$

3.6.3 AUC Value

AUC (Area Under ROC Curve) value calculates the area under the ROC curve. This is a numerical representation of the performance of a binary classifier, while the ROC curve is the visual representation of the performance of the said classifier, a sample ROC curve can be found in Figure 3.8. In the ROC curve plot, the x-axis is the FPR and the y-axis is the TPR.

In this thesis, the AUC value is used to evaluate the importance of a feature. The calculated AUC values are in the range of [0, 1], which can reflect the relevance between the feature and the predicted result to some extent. The higher the AUC value, the feature is more related to the predicted result. If the AUC is near 0.5, then this feature is no better than random guessing.

3.7 Proposed Thesis Approach

The goal of this research to discover the optimized machine learning model with the best-performing feature set for phishing website detection. To achieve this goal, I start my experiments with three stages, from data collection and preparation, to determine and improve the feature set and the machine learning model, and finally to optimize the model using the automatic machine learning tool, TPOT.

During the first stage, I design, build and evaluate the UCI and CANTINA+ feature sets on the existing UCI dataset as the baseline. Then new crawling, data collection and processing steps are performed to obtain around 9,000 phishing URLs and 1,500 legitimate URLs, both with the web page contents downloaded. This

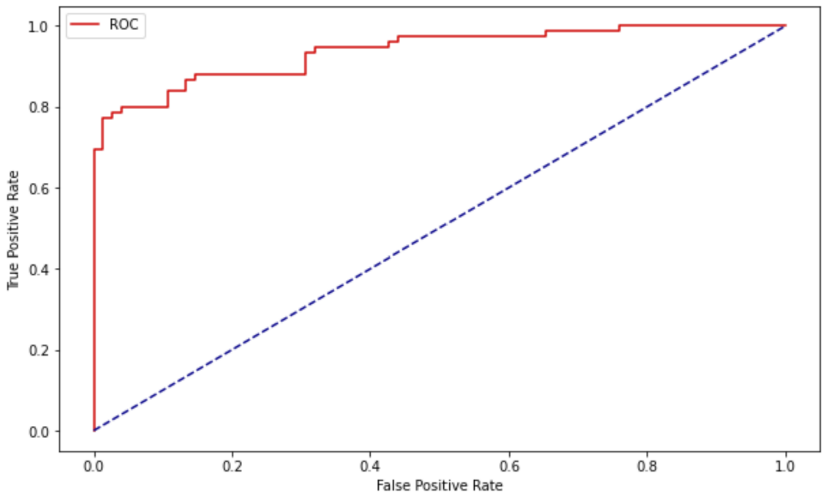

Figure 3.8: An example ROC curve plotted using Logistic Regression with random data points.

enabled the construction of three datasets from the URLs I collected, each with a different phishing/legitimate links ratio for testing purposes: Dataset-1 has a 1/1 phish/legit ratio, Dataset-2 has an 8/1 phish/legit ratio, and Phish-1 consists of phishing URLs only.

As I moved on to the next stage, I first build a baseline for the project using the existing dataset and feature sets from UCI and CANTINA+. At this stage, the performance of multiple machine learning classifiers, including Decision Tree, Random Forest, SVM, and so on are evaluated. As it will be presented in the following chapter of the thesis, the Decision Tree has the most stable and highest training and validation scores at this stage. I also analyze how the percentage of the number of phishing links in the dataset might affect the training results as well as analyze the features used by grouping them into subcategories. After extensive experiments on all the combinations of the candidate features, I identify and propose the best practice features achieving the highest prediction accuracy.

At the final stage, to optimize the detection model using the proposed feature set, I employ the automated machine learning tool, TPOT, which provides a performance boost on both the training and testing scores. The training process of TPOT only includes the training and validation dataset, which is Dataset-1 with 0.3 train_test_split, the testing dataset Dataset-2 is used to validate the model performance on unseen

data. The proposed approach is summarized in Figure 3.9. I then design and develop a web API that implements my best-performing model. The API can input any user given URL, analyze it using the proposed features, and report the prediction result in a near real-time manner.

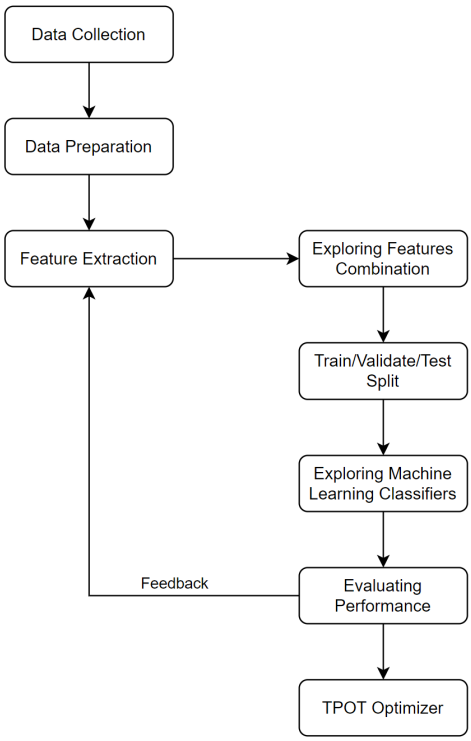

Figure 3.9: Overview of the proposed approach.

3.8 Summary

To summarize, the goal of my research is to present a machine learning model that could leveraged the best practice features yet still be able to provide the best possible performance that could be achieved on the given data. To this end, I analyze the URLs collected from many perspectives, based on the URL itself and based on the online third-party SEO tools. Moreover, text analysis is performed to observe the effect of the DOM object of a website and its effects on the phishing detection. This includes word frequency test TF-IDF and visualization tools such as word clouds. The detection model is in the form of a Decision Tree classifier, and optimized by automated ML tool, TPOT.

Chapter 4

Results and Discussion

In this section, I will be using tables and other visualization tools to describe the results of the evaluations and experiments performed. This starts with setting the baselines for this research using the state of the art approaches from the literature in terms of data, feature sets, and classifiers. Then, new features are added to improve upon the existing one and they are evaluated using new datasets as well as automating the optimization of the learning models.

All the scripts used in this research are coded in Python. To perform the machine learning classification tasks and evaluate the models, scikit learn libraries are used, including classifiers such as Logistic Regression, Decision Tree, and Random Forest. Moreover, evaluation metrics such as roc_curve(), f1_score() are also employed via scikit learn. To analyze the suspicious HTML documents in a secure mode, a Kali Virtual Machine environment with Debian 10.x 64-bit, 2GB RAM, 4 processors is built, as phishing websites tend to contain malicious contents.

4.1 Evaluations

As discussed in the literature review chapter, the existing literature [9] [22] uses UCI data and its corresponding feature set, as this is the most classic feature set, it forms the baseline in this thesis. As the dataset from the UCI online repository is encoded and the method used to determine the value for each feature is not documented well, Table 3.3 presents the method used in this work. Given this, the training and validation results using J48 Decision Tree classifier provided by Weka v3.8.4, with train_test_split of 0.33 and 10-fold cross-validation can be found in Table 4.1.

From the initial training results, it can be seen that the UCI feature set is compatible with new datasets collected in this thesis. The training F1-score for all datasets is around 90%. However, after increasing the size of the dataset, the performance when evaluating TPR and FPR for validation sets are dropped around 10%. As mentioned

Table 4.1: Baseline evaluations using UCI Feature Set

Dataset	F-Measure	TPR	FPR
UCI	0.901/0.890	0.901/0.891	0.076/0.091
Dataset-0	0.900/0.818	0.896/0.778	0.096/0.124
Dataset-1	0.894/0.814	0.854/0.809	0.191/0.244

* T is short for training, V is short for validation.

before, the difference between Dataset-0 and Dataset-1 is that the latter is crawled from the web pages in Dataset-0. The reason behind the drop in the performance might be due to the changing patterns of the linksin Dataset-1. To eliminate the influence, I shuffled the links in both datasets, and the results are encouraging, the F1-score for both datasets stays at 0.90, while the TPR is stable at around 0.89, and FPR is dropped back to 0.15.

4.2 Training Results Using Various Classifiers

With the baseline in mind, I moved to test the effect of the above feature set on various classification algorithms. Table 4.2 presents the results when training the model with classifiers Logistic Regression, KNN, SVM, AdaBoost, Decision Tree, Random Forest, Gaussian Process, Gaussian NB, and Quadratic Discriminant Analysis (QDA). The purpose for this experiment is to explore the possibilities of different classifiers, to evaluate if they are suitable for this use case. The parameters for each classifier are set via Grid Search CV with 10-fold cross-validation, the scoring method is set to "f1-micro", and as usual, the train_test_split is 0.33. The visualized tree can also be found in Figure 4.1. Please see Appendix A for evaluation results with more ML classifiers.

From these results, I can see that the Decision Tree classifier achieves the best performance with a training F1 score of 0.9689, while Random Forest with a training F1 score of 0.9500. However, the Random Forest classifier by definition is a collection of uncorrelated decision trees, with the results voted by the majority of trees. Thus, analyzing and visualizing the resulting classifier is difficult, the benefit of reducing the risk of overfitting does not overcome the complexity brought by the Random Forest.

Thus, I decided to continue with the Decision Tree classifier for the remaining

Table 4.2: Training/Validation Results with Dataset-1 and UCI features (without "F5 — Having IP Address" and "F9 — Web Traffic").

Classifier	F1score T/V	TPR T/V	FPR T/V
SVM	0.9130/0.9008	0.9437/0.9518	0.1212/0.1679
Decision Tree	0.9689/0.9065	0.9513/0.8965	0.0130/0.1131
Random Forest	0.9500/0.9027	0.9529/0.8960	0.0535/0.0878

* T is short for training, V is short for validation.

of the evaluations. To determine the importance of each feature, I also calculated the AUC value for the Decision Tree classifier. As presented in Figure 4.2, "F10 — Domain Age" affects the training greatly with 0.8318 AUC, this is expected as phishing websites tend to have shorter living time compared to regular websites. They have often been taken done quickly after being reported or detected. "F7 — Domain is IP Address" comes next, with 0.7373 AUC, I believe the reason behind this is that, due to how the data collection process proceeds, Alexa Top Sites will not admit a website with an IP address as hostname on their page rank list. Also, seldom legit sites would redirect their users to web pages with only an IP address and no other information provided in the URLs themselves. On the other hand, phishing URLs on PhishTank surely have no problem with using a single IPv4 address as their hostname.

Before I applied the CANTINA+ features to the Dataset-1, I first need to determine the threshold value for "F9 — Non-matching URLs". As explained in Chapter 3, this feature examines all the links embedded in the DOM object, it can be set to True if the appearance of any single URL pattern exceeds a given threshold. The pattern can be detected if the URLs have exactly the same content apart from the last layer of the path. Empty anchor tag references such as "#" fall under the same pattern. To find the value for the threshold, I trained the Decision Tree classifier with only the URL-based and HTML-based features on Dataset-1, with the threshold set to 30% and 50% respectively. The results are shown in Table 4.3, where with 50% threshold, the model has better F1-score and TPR, while the false positive is increased by merely 0.004. Thus, for the following experiments, I select the 50% as my threshold for this feature.

Table 4.3: Training results on Dataset-1 with URL-based and HTML-based features from CANTINA+ feature set with different threshold values for "F9 — Non-matching URLs"

Threshold	F1-score	TPR	FPR
30%	0.8162	0.856	0.263
50%	0.8372	0.862	0.267

When applying the CANTINA+ feature set to Dataset-1, the results achieved are given in Table 4.4. Similar to the UCI feature set, Decision Tree again is ranked at number one in the performance, while Random Forest came right after it. However, the overall training and validation results dropped compared to the UCI feature set. Note that, all the features applied in this case are the original features from the CANTINA+ paper [39] without being modified or removed. This seems to indicate that the features in the CANTINA+ feature set may not be suitable for new datasets.

The ROC value is again calculated for each feature, results are plotted in Figure 4.3. In this case, the web-based features "F13 — Page Rank" from Google and "F11 — Age of Domain" from Whois ranks the highest two with 0.8470 and 0.8244, respectively. "F9 — Non-matching URLs" surprisingly reached the number three with 0.7396 AUC. Overall, we observe that the web-based features tend to have greater impact on the model performance, as the results extracted from the Internet are harder to manipulate compared to the DOM object and the URL itself. Legitimate sites tend to have a better reputation on the web, and they would generally rank higher in search engine results. While the phishing sites are more likely to "hit-and-run", they would not leave many traces on the Internet, meaning shorter domain age and lesser Page Rank. Please see Appendix A for evaluation results with more ML classifiers.

Table 4.4: Training/Validation Results with Dataset-1 and CANTINA+ feature set.

Classifier	F1score T/V	TPR T/V	FPR T/V
SVM	0.9039/0.9025	0.9137/0.8907	0.0215/0.0891
Decision Tree	0.9682/0.8841	0.9676/0.8977	0.0012/0.1229
Random Forest	0.9583/0.9350	0.9305/0.9138	0.0105/0.0483

* Note: T is short for training, V is short for validation.

4.3 Phishing Links percentage in Training Dataset

Before proposing a new feature set, I need to determine if the current ratio between the phishing and the legitimate entries in the training dataset is appropriate or even has any effect on the detection. To this end, I vary the percentage of phishing entries in the training dataset from 90% to 10%. The training performance of the Decision Tree classifier in terms of F1-score and TPR can be found in Figure 4.4. As the percentage for phishing data increases, I can see that the training F1-score and TPR are also increased. With the highest F1 of 0.9836 at 90% and the lowest F1 of 0.8158 at 10%, the model gains a 17% improvement on F1. The trend shows in the figure proves that the model can detect more phishing websites with more patterns added to the training set. This result is also demonstrated by the experiment performed in the CANTINA+ paper [39]. To avoid being over-optimizing on a single aspect, I decided to set a phishing/legit ratio of 8:1. This is how I built my Dataset-2.

4.4 Proposing a new feature set: Categories of Features

Dividing the features into categories helps us to analyze them better. The chosen categories "URL-based", "HTML-based", and "Web-based" describe the nature of the features properly. The definition for each category can be described as below:

- URL-based Features. These are the features that can be extracted directly from the URL, e.g. the length of a URL, number of dots in the URL, or if the hostname of the URL is an IPv4 address.

- HTML-based Features. The features in this category require analyzing the DOM object of a website. Thus, to use and extract such features downloading the DOM object to process offline is necessary. Examples are observing the behaviors of the HTML form tags, or comparing the links embedded in the anchor tag.

- Web-based Features. These are the features that use the information extracted from the above two categories, such as the domain name or the most frequent words in the DOM object, combined with online resources for labeling. Examples could be using the domain name of a link to lookup its domain registration

data, Google search for the website's Copyright name and the hostname, checking if the top N search results' domain matches the target link's domain etc.

In this section, I experiment with the performance of multiple classifiers with using features from only one category at a time, and then compare them to the complete set. This allows us to understand how each category contributes to the overall performance. As the names of the categories suggest, the features belong to "URL-based" can be directly extracted from the link itself, while the "HTML-based" features require the researcher to observe the behavior of the DOM object, including summarizing the HTML tags and applying text analysis on the all the text contents inside the DOM object. "Web-based" features are collected by combining information extracted from the link locally and the online resources. Table 4.5 shows the F1-score for training the model with each feature category separately and the overall training results with multiple classifiers, Table 4.6 presents the TPR and FPR for the training results. Please see Appendix A for evaluation results with more ML classifiers.

Table 4.5: Training F1-score for each feature category.

Classifier	URL-based	HTML-based	Web-based	All Features
SVM	0.7573	0.8136	0.9216	**0.9704**
Decision Tree	0.8397	0.8154	0.9284	**0.9999**
Random Forest	0.8595	0.8123	**0.9425**	0.9842

Table 4.6: Training TPR and FPR for each feature category

Classifier	URL-based	HTML-based	Web-based	All Features
SVM	0.7117/0.1667	0.7990/0.1695	0.8977/0.0513	0.9639/0.0232
DT	0.7739/0.0713	0.8005/0.1618	0.8983/0.0360	0.9998/0.0011
RF	0.8194/0.0897	0.7976/0.1661	0.9373/0.0524	0.9759/0.0073

* Format of the data follows TPR/FPR.
** "DT" is short for Decision Tree, "RF" is short for Random Forest.

From the results of these evaluations, it could be observed that the "ALL Features" category reports the highest F1-score across all other categories. If one looks at an

individual category, then the "Web-based" category reports an overall 0.9 F1-score, offers an over 10% performance boost, while the "URL-based" and "HTML-based" features do not perform well on their own. Therefore, it seems that the combining these features improves the performance of the model. At the same time, it should be noted that the "Web-based" features have the highest priority. In the next section, I filter out the features to understand which ones are irrelevant, while setting the methods for each feature extracted with extensive experiments.

4.5 Finalizing the Proposed the Feature Set

In this phase, a training/validation dataset with 9,000 data instances (including 1,000 legitimate URLs and 8,000 phishing URLs, the ratio between legit/phish is 1/8) are employed, along with the DOM object for all the websites included in the dataset. The Decision Tree classifier is used with the CANTINA+ feature set. To explore more possible features, and to obtain an overview of DOM objects I have, I first summarized the file size distribution for both legit and phishing web pages, the results are shown in Figure 4.5. Legitimate pages have a gradually decreasing trend, around 30% of the legitimate pages have a size smaller than 95 kB without the image or video contents. While the distribution for phishing web pages is more extreme, over 95% of the pages have a size smaller than 187 kB, and an outlier file that has size greater than 1,688 kB.

Then, I extracted all the text messages for the DOM objects, applied the TF-IDF algorithm to search for the top five most frequent keywords for each document. I treated each DOM object as a document corpus, and the tags that contain text messages are treated as a single document. Combining the most frequent words with the domain name of the web page's URL, I have a string that could describe the web page. I used Google as my search engine as it provides a fast, online API to access the real-time search results. If the query successfully finds the domain in the first 20 search results, then I set the feature "Top Search" as True.

However, an issue was raised while I perform the TF-IDF algorithm, some of the web pages return a query that does not have any actual meaning. As I look at the actual page contents, I realized that some of the web pages, whether they are legitimate or phishing, only contain several keywords to guide their user to the

page that they are looking for. Thus, my most frequent word search would return words such as “login”, “signin”, “username”, that cannot be used as a search query in Google. Some shopping or web content illustration websites do contain more text, but they are rather a combination of keywords that help their user to locate certain goods than complete sentences.The most frequent words search in this case also failed to report useful information about the website. The percentage of such web pages could be rather high in real life and this is the case in the datasets I collected as well. In this thesis, this is roughly 50% of the total web pages, so I remove the “Top Search” feature not to bias the assessment.

Moreover, using features which require accessing the contents of an unknown and potentially malicious website could put a user (client) and the detection server in danger of being hacked. Additionally, the web page’s loading time and the processing time of the DOM object contributes additional delays in terms of computation time for the response time of a detection system that uses such features. With all these issues considered, I decided to filter out the HTML-based features entirely from the proposed feature set.

Thus, with only the “URL-based” and “Web-based” categories left, the list of the 17 candidate features that I propose for my phishing detection system are as the following:

- IP Address: If the domain name of the URL is an IP address or not.
- URL Length: Length of the URL.
- Shorten URL: If the URL uses any URL shortening services. A shortened URL is generally in the form of “domain of the shorten service/randomly generated string with characters and numbers”. Examples could be “http://bit.ly/35v5E4B”.
- Sub Domain: Number of subdomains in the URL’s hostname.
- Embedded Domain: If the path of the URL contains any three dots separated string segments that could be treated as a domain.
- Sensitive Words: If the URL contains any words in the sensitive word list. Current list is [“secure”, “account”, “webscr”, “login”, “ebayisapi”, “signin”,

"banking", "confirm"]. The new list is updated based on the most frequent words in my phishing datasets.

- Out-of-position TLD: If the URL contains any TLD other than the regular one.
- Domain Age: Days since the domain is registered.
- Top Search: Search in Google with the URL's domain name. If any of the first N returned results has a domain same as the URL's domain. In this research, N = 5.
- Page Rank: Results from Google's Page Rank service. The range for this feature is [0, 10], where websites with higher number values are the ones ranked higher in Google search.
- Number of Dots: Number of "." in the URL. An interesting observation I made while analyzing the links is that phishing URLs tend to use "dot" to replace ".", which might help them to escape some of the phishing detection mechanisms. Thus, for this feature, I also calculated the occurrence of the word "dot". The frequency for "." and "dot" are added to produce the final value for this feature.
- Number of Dashes: Number of "-" in the URL.
- Number of Underscores: Number of "_" in the URL.
- Number of At Symbols: Number of "@" in the URL.
- Number of Equal Symbols: Number of "=" in the URL.
- Number of Question Marks: Number of "?" in the URL.
- Number of Percent Marks: Number of "%" in the URL.

The current list of sensitive words in the proposed feature set is directly taken from the CANTINA+ Feature Set, according to the AUC values I measured over the evaluations reported above. This feature does not have much influence on the training results. The original list is constructed by Kang et al. [19] in 2007 using the eight words that were frequently occurring in phishing URLs in their datasets. To

improve this list and make it more adapted to the current data, I first measured the occurrence percentage of those eight words in both legitimate and phishing URLs. The results are shown in Table 4.7. In the table, I can see that the "webscr" has zero occurrences in all legitimate links, 0.0153% in all phishing links, does not match the description of "the most frequent words in phishing URLs".

Table 4.7: Default List of Sensitive Words Actual Frequency in All Legitimate and Phishing URLs

	Count	Percentage in all phishing URLs	Percentage in all legitimate URLs
"secure"	980	0.7136%	0.0352%
"account"	741	0.5396%	0.1126%
"webscr"	21	0.0153%	0.0000%
"login"	4301	3.1318%	0.2464%
"signin"	572	0.4165%	0.1689%
"banking"	76	0.0553%	0.1478%
"confirm"	159	0.1158%	0.0211%

To explore a better sensitive word list, I calculated the word frequency dot-separated string segments in both legitimate and phishing URLs, and plotted word clouds for demonstration purpose, the illustrations can be found in Figure 4.7 and Figure 4.6. The highlighted words in phishing URLs are "amp", "web", "login", "secure", where "amp" is frequently used when the link contains an "&" symbol in the URLs. "runescape" and "blogspot" are the most frequent domain names in the datasets used in this thesis. PhishTank has a large collection of phishing URLs from both domains, therefore, I do not use those domain names as they are not generalized enough. The distribution of words in legitimate links are rather even, I do not see many highlighted words as I saw in the word cloud of phishing links. "3A" and "2F" stand out in the word cloud, they could be representing the colon (":") and slash ("/") respectively in ASCII code for HTML entities. Investigating the reasons behind this is left to future work. The rest of the words with larger sizes are mostly the domain names for the legitimate URLs, such as "tmall", "forbes", "medium", "twitter", etc.

Therefore, the revised and improved list of sensitive words now consists of two parts: words in the original list that have a higher percentage in phishing URLs

according to Table 4.7, and the words that I discovered with word frequency calculation and the phishing word clouds that I generated for the most up to date data employed in this thesis. The final list includes “secure”, “account”, “login”, “signin”, “confirm”, “amp”, “web”.

To reach the best-performed model with minimized features, I start with training the model using Decision Tree Classifier with all the features listed above, remove one feature at a time for each epoch, until I obtained a balanced result with minimum features in the set while still has a well training/validating performance.Table 4.8 presents the training and validation results when using all the candidate features.

Table 4.8: Training results with all 17 proposed features using grid search Decision Tree

	F1-score	TPR	FPR
Training	0.9887	0.9908	0.0876
Validating	0.9760	0.9805	0.2005

Table 4.9 shows a sample combination set of URL-based only features trained and validated using Decision Tree classifier (GridSearchCV). The values (variations) in the list are “F3 — Prefix/Suffix”, “F7 — Sensitive Words”, “F9 — Special Symbols”, the default value is “A1, B1, C1” when:

- A1 (F3 — Prefix/Suffix): If a URL’s hostname contains “-” or “_”
- B1 (F7 — Sensitive Words): If a URL contains any of the keywords in “secure”, “account”, “webscr”, “login”, “signin”, “banking”, ’confirm’.
- C1 (F9 — Special Symbols): If a URL contains “@” or “_”

The values of the sample features are “A2, B2, C2”, when:

- A2 (F3 — Prefix/Suffix): If a URL’s hostname contains “-” or “_”
- B2 (F7 — Sensitive Words): If a URL contains any of the keywords in “secure”, “account”, “webscr”, “login”, “signin”, “banking”, ’confirm’, **“dot”**.
- C2 (F9 — Special Symbols): If a URL contains “@”

Table 4.9: Samples of Combination of Features with URL-based only Training/Validation Results

Features	F1-score (T/V)	TPR (T/V)	FPR (T/V)
A1, B1, C1	0.7833/0.7440	0.7105/0.6720	0.1030/0.1359
A2, B1, C1	0.8229/0.7541	0.8616/0.7957	0.2315/0.3179
A1, B2, C1	0.7790/0.7439	0.6907/0.6559	0.0822/0.1087
A1, B1, C2	0.7849/0.7440	0.7128/0.6720	0.1030/0.1359
A2, B2, C1	0.8217/0.7494	0.8547/0.7876	0.2245/0.3179
A1, B2, C2	0.7806/0.7431	0.6930/0.6532	0.0822/0.1059

From the results above, I chose to keep the combination of “”A2, B1, C1”, which has the highest F1-score and TPR for both training and validation experiments. The FPR for this combination is higher than “A2, B2, C1” by 0.007, but the overall performance is still at the peak. Additional training/validation results can be found in Appendix A.

4.6 Comparison Between Grid Search Decision Tree and TPOT Pipeline

After extensive experiments on the features, I propose the above 17 features for representing a website data to a phishing detection classifier, namely the decision tree. The next step is to test the decision tree model trained using the proposed features on the unseen Dataset-2.

The generalization of the classifier over time is one of the issues evaluated in this section. In order to observe if the proposed feature set can still correctly detect phishing websites after a time period passed since the training data collection, I use the Dataset-2 for testing, which is downloaded six months after the training data collection stage. The results look promising. For grid search Decision Tree, the validation result has lower TPR and higher FPR compared to the performance observed during training. The F1-score is dropped by 2%. As for the testing results, F1-score dropped by 4% compared to training, TPR and FPR also dropped, but the difference seems to be reasonable. Considering that testing data is newer than training data and the legitimate links in the Dataset-2 are collected without using the Alexa Top Site service.

Finally, training the TPOT pipeline on Dataset-2 will then help us obtain a fitted

model for testing. In this use case, I have the training F1-score as 0.9565, validation score as 0.8986 and testing F1 as 0.8200. All the scoring metrics show a higher accuracy compared to the traditional Decision Tree classifier. Table 4.10 presents the comparison results between TPOT optimization and Decision Tree grid search tuning for training and testing datasets, and Figure 4.8 presents a complete generation of the TPOT optimizer, including the pre_test decorator and the training classifiers. For the TPOT optimizer, the best pipeline for classification XGB classifier, and as expected, I received a performance boost when comparing to the grid search Decision Tree. The validation result of F1-score for TPOT classification is 0.98 and the testing F1-score is approximately 0.96.

Table 4.10: Comparison between TPOT and Decision Tree classifier

	F1-score	TPR	FPR
DT (Train)	0.9679	0.9647	0.1902
DT (Validate)	0.9484	0.9506	0.2079
DT (Test)	0.9274	0.9203	0.2910
TPOT Class (Train)	0.9908	0.9942	0.0694
TPOT Class (Validate)	0.9832	0.9858	0.1355
TPOT Class (Test)	0.9523	0.9433	0.2801

* "DT" is short for Decision Tree
** "Class" is short for classification

4.7 Web API for Users

Last bit not the least, I designed and developed a web API that is based on the proposed feature set using the best training model. The web page is rendered in Flask, which is a Python web framework with easy-to-use libraries and modules. Flask is based on Werkzeg's routing, debugging, and WSGI (Web Server Gateway Interface), along with Jinja2 as the template engine support [14]. The website is currently running on localhost and is possible to go online if needed.

To minimize the response time when a user makes a request, I choose to keep all the machine learning processes on the local server so that I do not have to redo an online training session every time. To archive this goal, I serialize the optimized model

as a .sav file using joblib library, load and restore the model when I need to make a new prediction. The aim has been to bring down the feature extraction process to a near real-time to minimize the response time for the user quesry. I addressed this by minimizing the number of features and easing the extraction of them for the training model in the first place. Furthermore, I also need to remove the HTML-based features that require loading the target website, which increases the response time and potentially put the user and my server in danger of cyber attacks.

The user interface of the API is simple and straightforward: The user enters the full URL of the website that they would like to check, the API will then return the predicted legitimacy of the link, along with the detailed results of the features that were used to test the link. A sample test case is shown in Figure 4.9, with the testing URL of "https://www.dal.ca/", the predicted result is "Legitimate", which is correct!

4.8 Summary

To summarize, in this section I presented the training and validation results on the Dataset-1 with multiple machine learning classifiers, including Decision Tree, Random Forest, SVM and others. I also discussed how the percentage of phishing data entries in the training dataset might affect the model performance, with a line graph to demonstrate my findings. After extensive experiments on different combinations of the proposed features, I achieved the set where I could use the minimum number of features yet can still achieve a well-performing phishing detection model. I compare the results of grid search Decision Tree, which is the best-performing classifier among others, and automated TPOT pipeline. To this end, the training and testing results show that with the optimized pipeline, TPOT could achieve an additional 2% boost in F1-score and 3% in TPR. The FPR of the TPOT results remains similar to the grid search FPR, both are around 0.2. Finally, I utilize the final ML model with a web API that could classify a user-provided URL link as phishing or legit in near real-time. The prediction accuracy rate of the API inherits the performance of the optimized model using the proposed feature sert, along with a straightforward user interface, the prototype is ready for use and will be made public after the complation of the thesis.

```
url_length = 0: -1 (307.0)
url_length = 1
|   url_anchor = -1: 1 (422.0/31.0)
|   url_anchor = 0
|   |   ip_address = 0: 1 (14.0)
|   |   ip_address = 1
|   |   |   domain_age = -1: 1 (17.0/6.0)
|   |   |   domain_age = 0: -1 (8.0/2.0)
|   |   |   domain_age = 1: 1 (4.0)
|   |   ip_address = 2: -1 (12.0/4.0)
|   |   ip_address = 3
|   |   |   domain_age = -1: -1 (7.0)
|   |   |   domain_age = 0: -1 (0.0)
|   |   |   domain_age = 1: 1 (5.0)
|   |   ip_address = 4: 1 (1.0)
|   url_anchor = 1
|   |   prefix = 0
|   |   |   sfh = -1: 1 (10.0)
|   |   |   sfh = 0
|   |   |   |   ip_address = 0: -1 (0.0)
|   |   |   |   ip_address = 1: -1 (91.0/27.0)
|   |   |   |   ip_address = 2
|   |   |   |   |   sub_domain = 0
|   |   |   |   |   |   domain_age = -1: -1 (11.0/2.(
|   |   |   |   |   |   domain_age = 0: 1 (1.0)
|   |   |   |   |   |   domain_age = 1: 1 (5.0/1.0)
|   |   |   |   |   sub_domain = 1: -1 (5.0)
|   |   |   |   |   sub_domain = 2: -1 (0.0)
|   |   |   |   |   sub_domain = 3: -1 (0.0)
|   |   |   |   |   sub_domain = 4: -1 (0.0)
|   |   |   |   ip_address = 3
|   |   |   |   |   domain_age = -1: -1 (4.0/1.0)
|   |   |   |   |   domain_age = 0: -1 (3.0/1.0)
|   |   |   |   |   domain_age = 1: 1 (6.0)
|   |   |   |   ip_address = 4: -1 (0.0)
|   |   |   sfh = 1: -1 (30.0/7.0)
|   |   |   sfh = 2: 1 (9.0/1.0)
|   |   prefix = 1: -1 (29.0/1.0)

Number of Leaves  :     28

Size of the tree :      39
```

Figure 4.1: Visualization for the Decision Tree generated with UCI features.

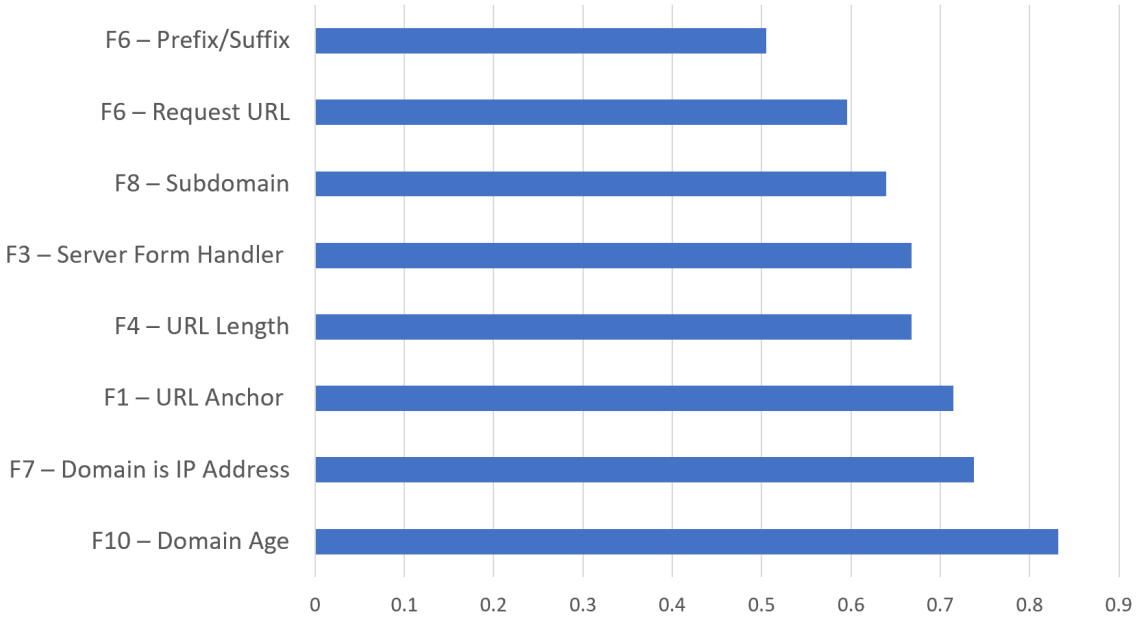

Figure 4.2: Area Under ROC Curve (AUC) bar plot for UCI feature set with Random Forest training results.

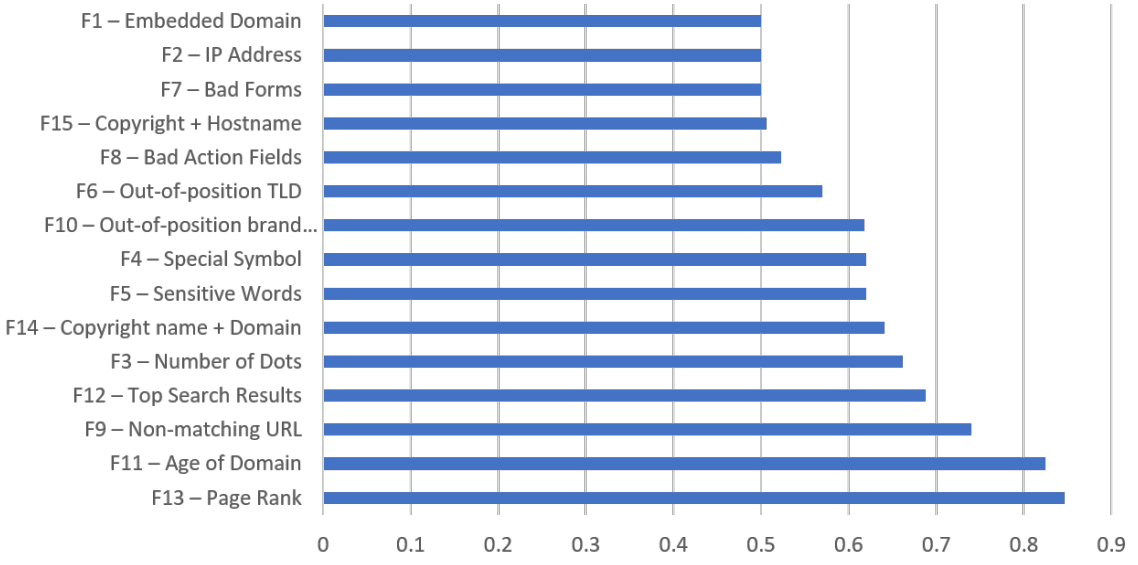

Figure 4.3: Area Under ROC Curve (AUC) bar plot for CANTINA+ feature set with Random Forest training results.

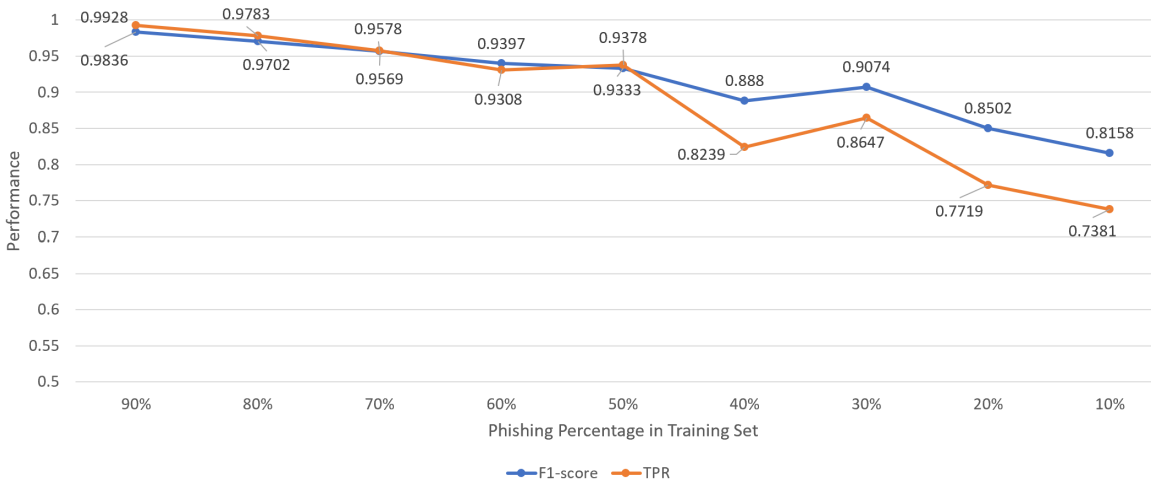

Figure 4.4: Phishing links percentages vs. Training F1-score and TPR.

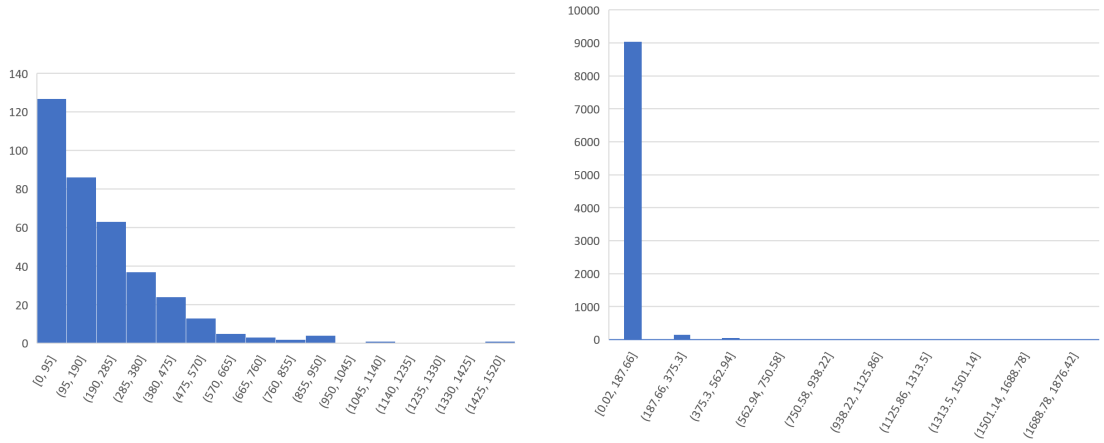

Figure 4.5: File size in kB for the HTML documents of legitimate websites (left) and phishing websites (right).

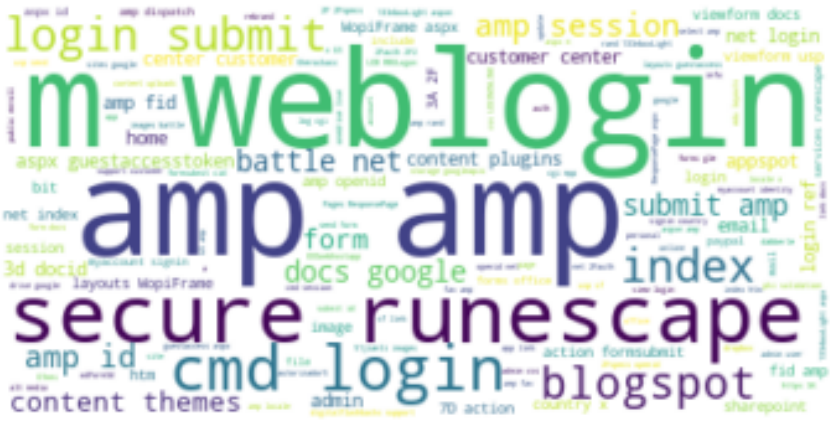

Figure 4.6: Wordcloud for word phases in all phishing URLs.

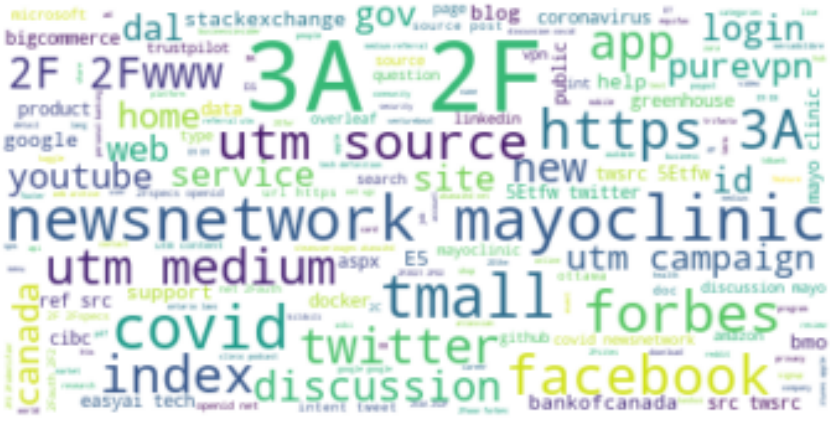

Figure 4.7: Wordcloud for word phases in all phishing URLs.

```
Generation 1 - Current Pareto front scores:

-1      0.9580985432703886      RandomForestClassifier(input_matrix, RandomForestClassifier__bootstrap=True, RandomForestCla
ssifier__criterion=gini, RandomForestClassifier__max_features=0.8, RandomForestClassifier__min_samples_leaf=2, RandomForestC
lassifier__min_samples_split=16, RandomForestClassifier__n_estimators=100)

-2      0.9593131951381665      GradientBoostingClassifier(MaxAbsScaler(input_matrix), GradientBoostingClassifier__learning_
rate=0.1, GradientBoostingClassifier__max_depth=4, GradientBoostingClassifier__max_features=0.9500000000000001, GradientBoos
tingClassifier__min_samples_leaf=9, GradientBoostingClassifier__min_samples_split=9, GradientBoostingClassifier__n_estimator
s=100, GradientBoostingClassifier__subsample=1.0)
Saving periodic pipeline from pareto front to tpot_mnst1.txt\pipeline_gen_1_idx_0_2021.04.12_09-09-42.py
Saving periodic pipeline from pareto front to tpot_mnst1.txt\pipeline_gen_1_idx_1_2021.04.12_09-09-42.py
_pre_test decorator: _random_mutation_operator: num_test=0 (slice(None, None, None), 0).
_pre_test decorator: _random_mutation_operator: num_test=0 Negative values in data passed to MultinomialNB (input X).
_pre_test decorator: _random_mutation_operator: num_test=0 Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penal
ty..
_pre_test decorator: _random_mutation_operator: num_test=0 Unsupported set of arguments: The combination of penalty='l1' and
loss='squared_hinge' are not supported when dual=True, Parameters: penalty='l1', loss='squared_hinge', dual=True.
_pre_test decorator: _random_mutation_operator: num_test=1 Unsupported set of arguments: The combination of penalty='l1' and
loss='hinge' is not supported, Parameters: penalty='l1', loss='hinge', dual=True.
_pre_test decorator: _random_mutation_operator: num_test=0 Unsupported set of arguments: The combination of penalty='l2' and
loss='hinge' are not supported when dual=False, Parameters: penalty='l2', loss='hinge', dual=False.
_pre_test decorator: _random_mutation_operator: num_test=0 Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penal
ty..
_pre_test decorator: _random_mutation_operator: num_test=0 feature_names mismatch: ['0', '1', '2', '3', '4', '5', '6'] ['f
0', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6']
expected 3, 4, 6, 0, 2, 1, 5 in input data
training data did not have the following fields: f4, f1, f3, f6, f2, f0, f5.
_pre_test decorator: _random_mutation_operator: num_test=0 (slice(None, None, None), 0).
_pre_test decorator: _random_mutation_operator: num_test=0 Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penal
ty..
```

Figure 4.8: A complete generation of the TPOT optimizer.

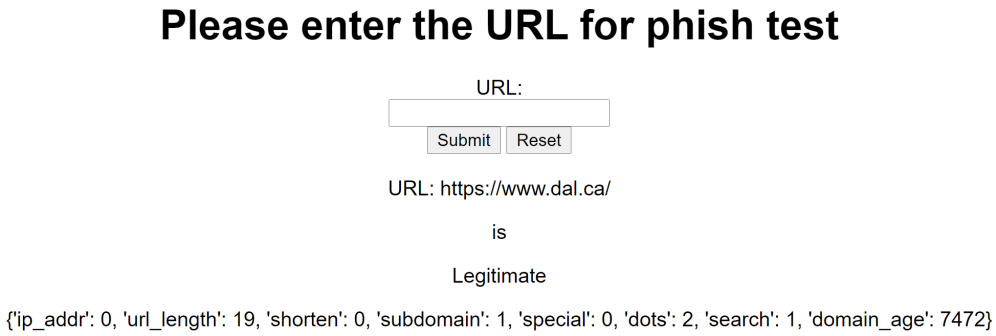

Figure 4.9: Screenshot when using the web API to predict the legitimacy of "https://www.dal.ca/".

Chapter 5

Conclusion and Future Work

The goal of my research is to obtain a machine learning model that could detect the phishing URLs with easy-to-extract features. In order to achieve this target, I start with experimenting with existing feature sets, both the UCI feature set and the CANTINA+ feature set. After obtaining a baseline model training and evaluation, I moved on to generate the best practice feature set by testing on all the combinations of features, and evaluating the model based on the performance it reaches on the testing set. At the last stage of my research, I feed the proposed feature set to the TPOT training pipeline to obtain the optimum model. I also built a web API that could load my TPOT model and report the prediction result to a user query in an online form.

The proposed feature-based machine learning model for phishing website detection can be easily implemented with only the target URL and Internet connection. As this topic has been a popular research interest, many studies prior to my research has been proven successful with high accuracy. My new contribution to this topic is to discover the best practice feature set with a small number of easy-to-extract features, trained and validated on the classifier algorithm that has the best performance optimized by automated machine learning toolkit. Evaluations employed datasets from different time periods with unseen new data and the results are reported using the F1-score, TPR and FPR metrics.

From the results of the experiments, when training with the scikit learn Decision Tree classifier, I obtained a F1-score of 0.9694, TPR of 0.9521, and FPR of 0.0957, while with the TPOT automated machine learning pipeline, I boost the accuracy of the proposed model to 96.99%. Compared to the results reported in the CANTINA+, which has 0.9225 TPR, 1.375 FPR and 0.9529 F1-score [39]. I can conclude that I have achieved my goal.

I also designed and developed a web API that could make use of my model. The

average processing time for the API is within 10 seconds, and it does not require the users of my API to access the suspicious website themselves. This provides a sense of security while still being able to accurately learn the legitimacy of an unknown website.

However, as every research there are limitations to my model. It can be improved by further studying other datasets and training approaches. At the beginning of my research, I collected over 9000 phishing URLs and over 1000 legitimate URLs. For future studies, I would like to enlarge the size of my dataset to more than ten thousand. A large sample size might also increase the model's prediction accuracy, added more reliability to the conclusion I draw from the experiments. I would also like to involve more data sources for both legitimate and phishing websites, currently, I am only using phishing links provided by PhishTank, and legitimate URLs from Alexa Top Sites and other official websites. Kaggle is another great source for data collection. It involves more data sources and therefore could provide more patterns of websites. Thus, could be explored as the next step of future work for generalziation purposes.

There are several other automated machine learning tools released after I started my experiments. One such tool is Google's Model Search, which is designed for model architecture search at scale. The Model Search system consists of multiple trainers, a search algorithm, a transfer learning algorithm and a database to store the various evaluated models [21]. The trainers will run independently to train and evaluate a model, share the results with the search algorithm. The search algorithm will then mutate the best architecture, feedback to the trainers for the next generation of training. An illustration from the Google AI Blog can be found in Figure 5.1, which demonstrates the procedure of the Model Search algorithm. In comparison, the TPOT classifier which is used in my research has the default number of generations of 100, the runtime for a TPOT classifier to finish searching might take up to ten minutes, and this is only with a limited amount of data points. Model Search, on the other hand, can run a distributed async search, which could be more efficient when processing large datasets.

Deep learning and Neural Networks are also strong classifiers that could be used in the future work. Thus, it is another area of interest that could be explored. Lastly, the

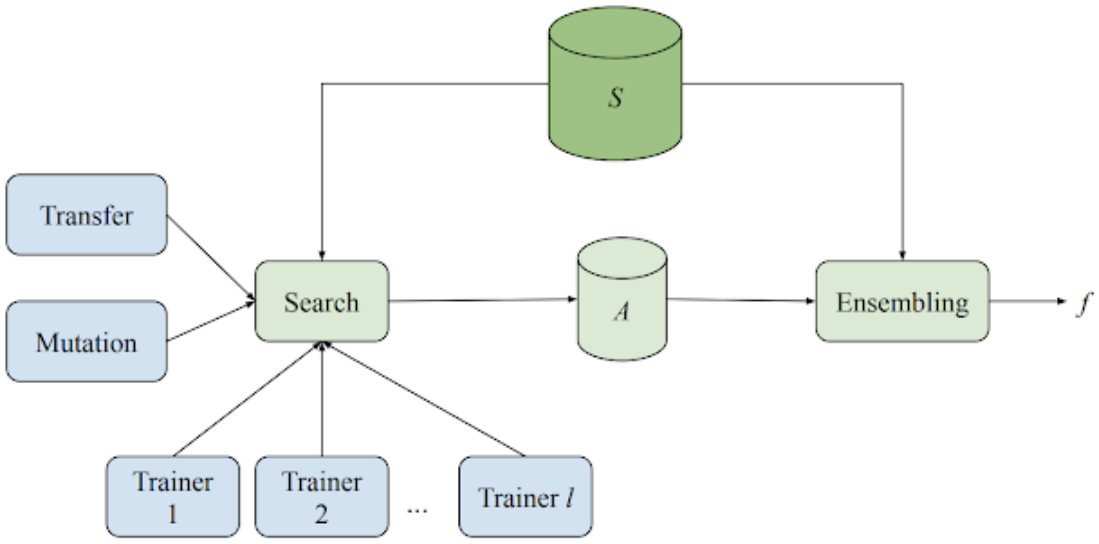

Figure 5.1: An illustration of the Model Search Working Algorithm. [21].

proposed model is based on features extracted from the URL itself and contribution from the SEO tools. For future work, the analysis on the network activities behind the phishing websites could also be explored to detect if the website is trying to install malware or malicious software on the victim's computer.

Bibliography

[1] Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. Phishing detection based associative classification data mining. *Expert Systems with Applications*, 41(13):5948–5959, 2014.

[2] Moruf A Adebowale, Khin T Lwin, Erika Sanchez, and M Alamgir Hossain. Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. *Expert Systems with Applications*, 115:300–313, 2019.

[3] Adithya Balaji and Alexander Allen. Benchmarking automatic machine learning frameworks. *arXiv preprint arXiv:1808.06492*, 2018.

[4] Simon Bell and Peter Komisarczuk. An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–11, 2020.

[5] Hong Bo, Wang Wei, Wang Liming, Geng Guanggang, Xiao Yali, Li Xiaodong, and Mao Wei. A hybrid system to find & fight phishing attacks actively. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 506–509. IEEE, 2011.

[6] Ahmet Selman Bozkir and Murat Aydos. Logosense: A companion hog based logo detection scheme for phishing web page and e-mail brand recognition. *Computers & Security*, 95:101855, 2020.

[7] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[8] Cisco. Email: Click with caution. Technical report, Cisco, 2019.

[9] Alfredo Cuzzocrea, Fabio Martinelli, and Francesco Mercaldo. A machine-learning framework for supporting intelligent web-phishing detection and analysis. In *Proceedings of the 23rd International Database Applications & Engineering Symposium*, pages 1–3, 2019.

[10] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. *URL http://papers. nips. cc/paper/5872-efficient-and-robust-automated-machine-learning*, 2015.

[11] Yasuhiro Fujiwara, Yasutoshi Ida, Sekitoshi Kanai, Atsutoshi Kumagai, Junya Arai, and Naonori Ueda. Fast random forest algorithm via incremental upper bound. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2205–2208, 2019.

[12] Pieter Gijsbers, Joaquin Vanschoren, and Randal S Olson. Layered tpot: Speeding up tree-based pipeline optimization. *arXiv preprint arXiv:1801.06007*, 2018.

[13] Google. Google Safe Browsing. https://safebrowsing.google.com/. (accessed: 03.30.2021).

[14] Miguel Grinberg. *Flask Web Development: Developing Web Application with Python*. O'Reilly Media, Inc., 2018.

[15] Anti-Phishing Working Group. About us. https://apwg.org/about-us/. (accessed: 04.14.2021).

[16] H2O.ai. H2O Documentation - Algorithms. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science.html. (accessed: 04.14.2021).

[17] Jason Hong. The state of phishing attacks. *Commun. ACM*, 55(1):74–81, January 2012.

[18] IBM. Cics transaction server for z/os - the components of a url. https://www.ibm.com/docs/en/cics-ts/5.1?topic=concepts-components-url. (accessed: 04.01.2021).

[19] Min Gyung Kang, Pongsin Poosankam, and Heng Yin. Renovo: A hidden code extractor for packed executables. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode*, WORM '07, page 46–53, New York, NY, USA, 2007. Association for Computing Machinery.

[20] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated Machine Learning*, pages 81–95. Springer, Cham, 2019.

[21] Hanna Mazzawi and Xavi Gonzalvo. Introducing model search: An open source platform for finding optimal ml models. https://ai.googleblog.com/2021/02/introducing-model-search-open-source.html?m=1. (accessed: 04.01.2021).

[22] Mahmood Moghimi and Ali Yazdian Varjani. New rule-based phishing detection method. *Expert systems with applications*, 53:231–242, 2016.

[23] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing, 2016.

[24] OpenPhish. OpenPhish FAQ. https://openphish.com/faq.html. (accessed: 03.30.2021).

[25] AA Orunsolu, AS Sodiya, and AT Akinwale. A predictive model for phishing detection. *Journal of King Saud University-Computer and Information Sciences*, 2019.

[26] Vaibhav Patil, Pritesh Thakkar, Chirag Shah, Tushar Bhat, and SP Godse. Detection and prevention of phishing websites using machine learning approach. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pages 1–5. IEEE, 2018.

[27] Gordon Paynter, Len Trigg, Eibe Frank, and Richard Kirkby. Attribute-Relation File Format (ARFF). https://www.cs.waikato.ac.nz/ml/weka/arff.html. (accessed: 03.30.2021).

[28] PhishTank. PhishTank FAQ. http://phishtank.org/faq.php. (accessed: 03.30.2021).

[29] David Prantl and Martin Prantl. Website traffic measurement and rankings: competitive intelligence tools examination. *International Journal of Web Information Systems*, 2018.

[30] R Rajalakshmi. Identifying health domain urls using svm. In *Proceedings of the Third International Symposium on Women in Computing and Informatics*, pages 203–208, 2015.

[31] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais. Phishdump: A multi-model ensemble based technique for the detection of phishing sites in mobile devices. *Pervasive and Mobile Computing*, 60:101084, 2019.

[32] Ina Salihovic, Haris Serdarevic, and Jasmin Kevric. The role of feature selection in machine learning for detection of spam and phishing attacks. In *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*, pages 476–483. Springer, 2018.

[33] scikit learn. Decision trees user guide. https://scikit-learn.org/stable/modules/tree.html. (accessed: 04.14.2021).

[34] Sifatullah Siddiqi and Aditi Sharan. Keyword and keyphrase extraction techniques: a literature review. *International Journal of Computer Applications*, 109(2), 2015.

[35] Choon Lin Tan, Kang Leng Chiew, Kelvin SC Yong, Johari Abdullah, Yakub Sebastian, et al. A graph-theoretic approach for the detection of phishing webpages. *Computers & Security*, 95:101793, 2020.

[36] Kiran D Tandale and Sunil N Pawar. Different types of phishing attacks and detection techniques: A review. In *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, pages 295–299. IEEE, 2020.

[37] Jonathan Tang. To Break Google's Monopoly on Search, Make Its Index Public. https://news.ycombinator.com/item?id=20440079. (accessed: 04.01.2021).

[38] Verizon. 2018 data breach investigations report. Technical report, Verizon, 2018.

[39] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+ a feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):1–28, 2011.

Appendix A

Figures

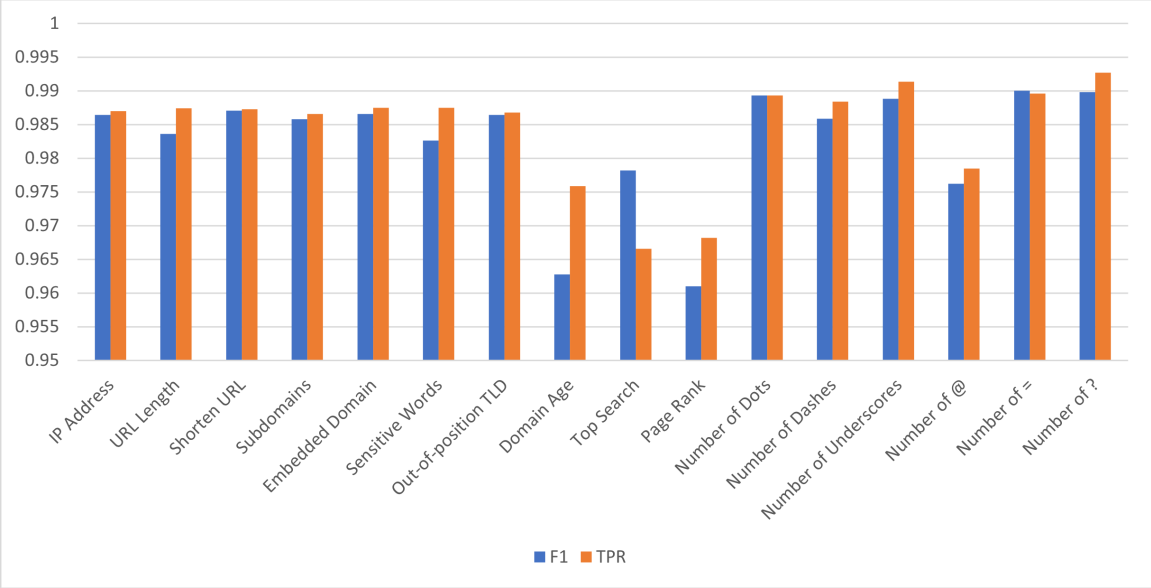

Figure A.1: Validation results' F1-score and TPR for Generation 1 with grid search Decision Tree

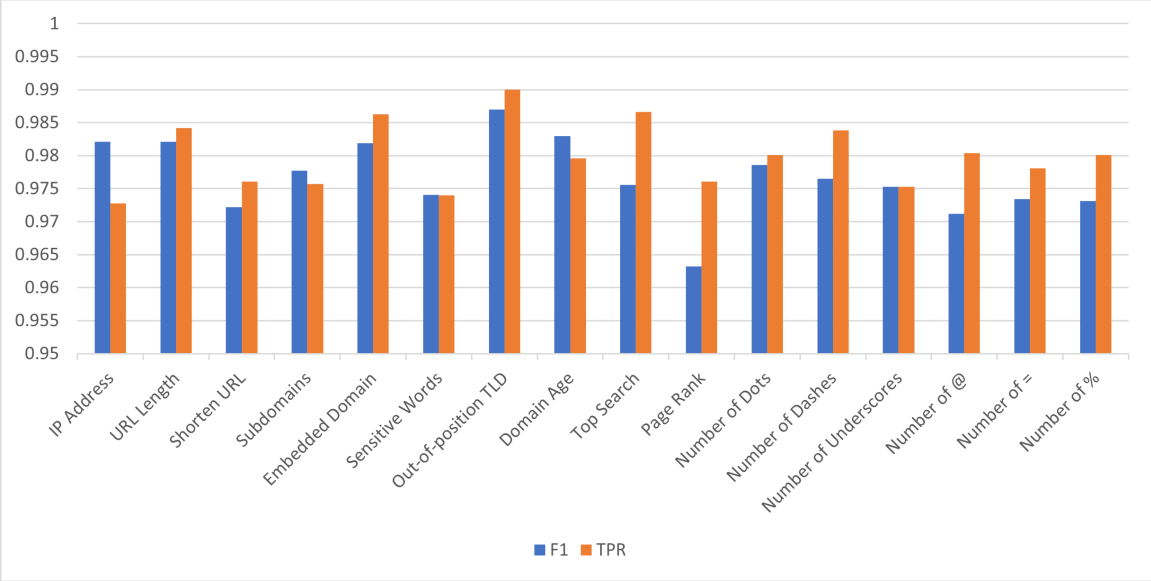

Figure A.2: Validation results' F1-score and TPR for Generation 2 with grid search Decision Tree, with "Number of ?" removed. The removed feature has the highest validation F1-score and TPR when removing the feature from the set.

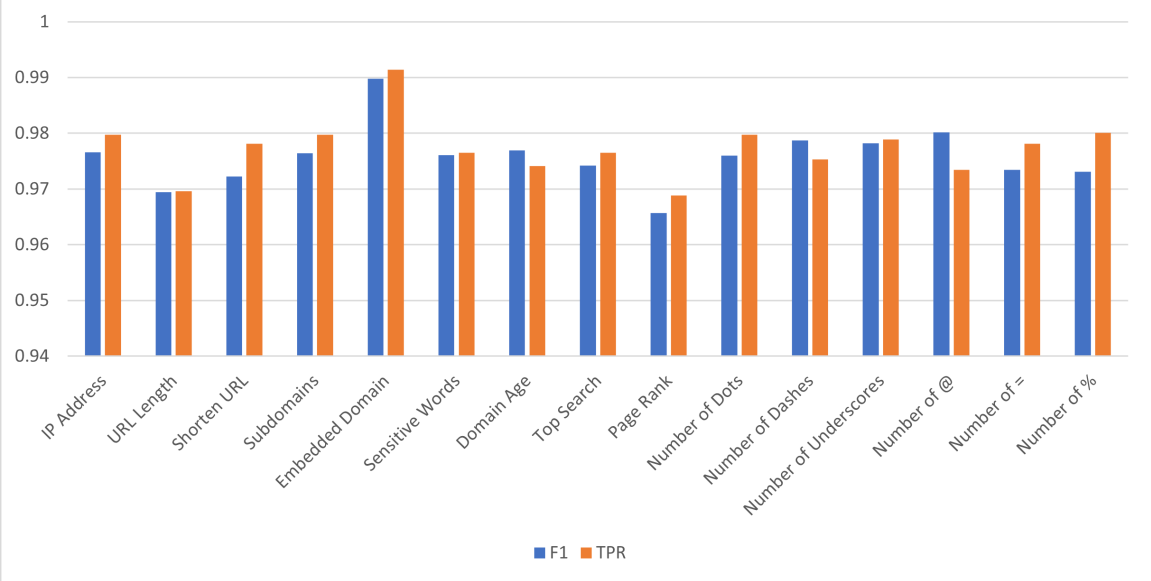

Figure A.3: Validation results' F1-score and TPR for Generation 3 with grid search Decision Tree, with "Out-of-position TLD" removed

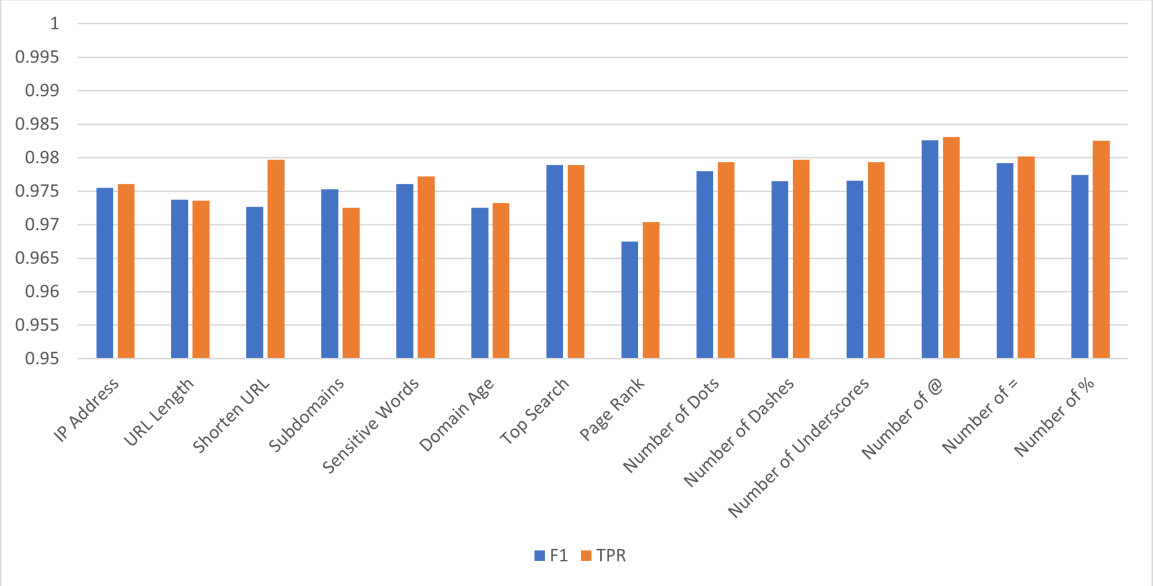

Figure A.4: Validation results' F1-score and TPR for Generation 4 with grid search Decision Tree, with "Embedded Domain" removed

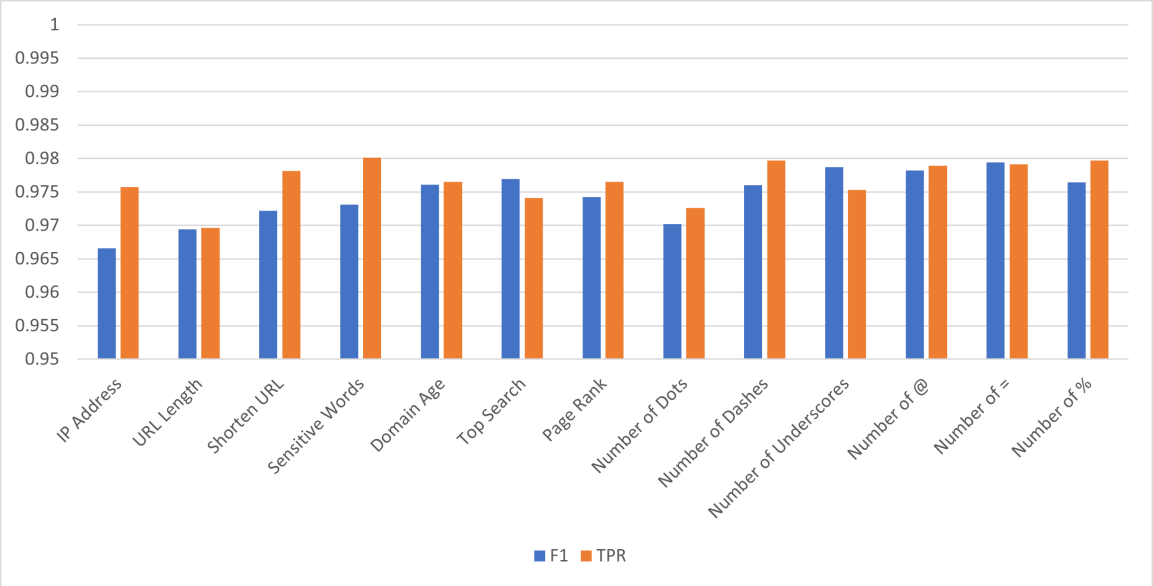

Figure A.5: Validation results' F1-score and TPR for Generation 5 with grid search Decision Tree, with "Subdomains" removed

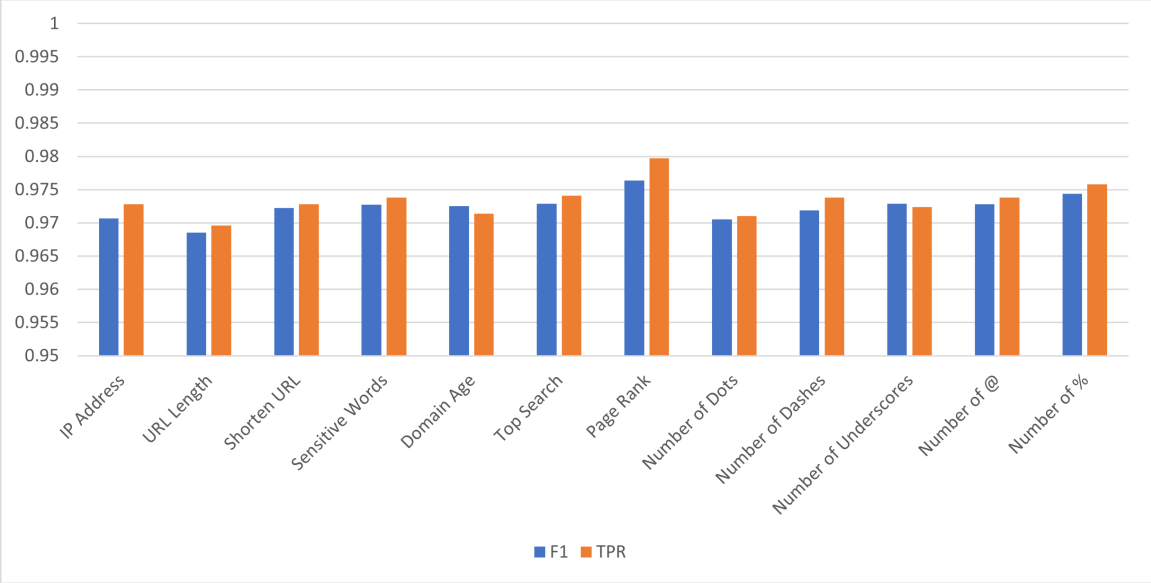

Figure A.6: Validation results' F1-score and TPR for Generation 6 with grid search Decision Tree, with "Number of =" removed.

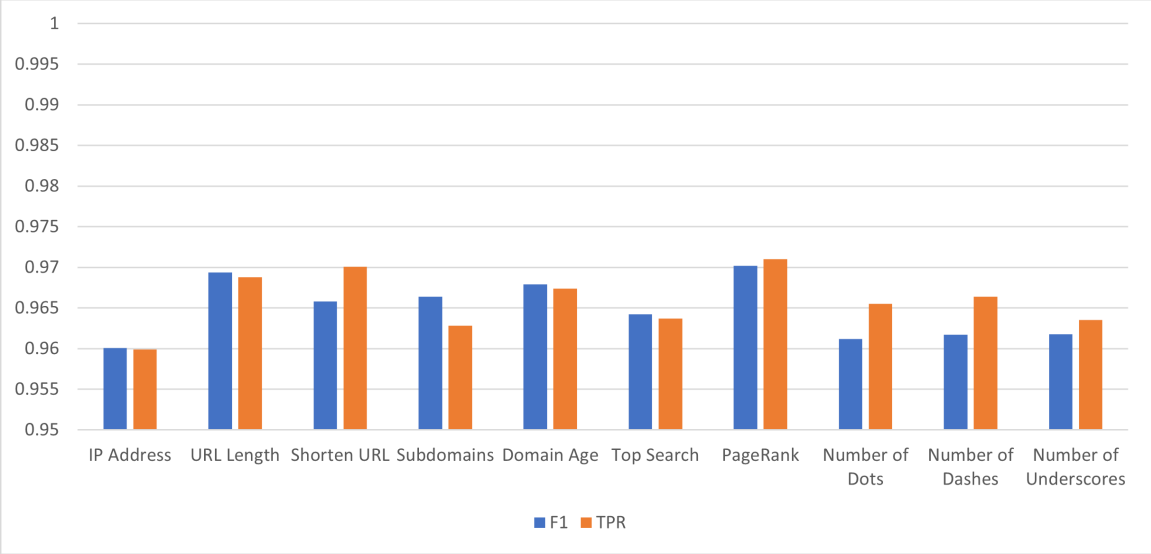

Figure A.7: Validation results' F1-score and TPR for Generation 7 with grid search Decision Tree, with "PageRank" removed. The process stopped as we reached feature number eight, both F1-score and TPR of the model dropped below 0.97, which is the threshold we set for this process.

Table A.1: Training/Validation Results with Dataset-1 and UCI features (without "F5 — Having IP Address" and "F9 — Web Traffic").

Classifier	F1score T/V	TPR T/V	FPR T/V
Logistic Regression	0.8817/0.8849	0.9205/0.9279	0.1695/0.1650
KNN	0.9150/0.8832	0.9242/0.8447	0.1006/0.1221
SVM	0.9130/0.9008	0.9437/0.9518	0.1212/0.1679
Ada Boost	0.9256/0.9177	0.9430/0.9312	0.0946/0.0983
Decision Tree	0.9689/0.9065	0.9513/0.8965	0.0130/0.1131
Random Forest	0.9500/0.9027	0.9529/0.8960	0.0535/0.0878
Gaussian Process	0.9106/0.8981	0.9431/0.9335	0.1286/0.1446
Gaussian NB	0.3756/0.3412	0.2352/0.2089	0.017/0.0123
QDA	0.8712/0.8551	0.8901/0.8688	0.1545/0.1610

* Note: "T" is short for training, "V" is short for validation.

Table A.2: Training/Validation Results with Dataset-1 and CANTINA+ feature set.

Classifier	F1score T/V	TPR T/V	FPR T/V
Logistic Regression	0.8698/0.8654	0.8329/0.8230	0.0801/0.0833
KNN	0.9411/0.8908	0.9286/0.8744	0.0449/0.0882
SVM	0.9039/0.9025	0.9137/0.8907	0.0215/0.0891
Ada Boost	0.9337/0.9156	0.9325/0.9179	0.0659/0.0850
Decision Tree	0.9682/0.8841	0.9676/0.8977	0.0012/0.1229
Random Forest	0.9583/0.9350	0.9305/0.9138	0.0105/0.0483
Gaussian Process	0.9476/0.9021	0.9368/0.8906	0.0419/0.0784
Gaussian NB	0.5837/0.6111	0.4281/0.4617	0.0389/0.0489
QDA	0.8492/0.8647	0.7526/0.7840	0.0251/0.0251

* Note: "T" is short for training, "V" is short for validation.

Table A.3: Training F1-score for each feature category using multiple classifiers

Classifier	URL-based	HTML-based	Web-based	All Features
Logistic Regression	0.6948	0.7986	0.8390	0.9067
KNN	0.7860	0.8203	0.9273	**0.9999**
SVM	0.7573	0.8136	0.9216	**0.9704**
Ada Boost	0.7948	0.8014	0.8930	**0.9599**
Decision Tree	0.8397	0.8154	0.9284	**0.9999**
Random Forest	0.8595	0.8123	**0.9425**	0.9842
Gaussian Process	0.7694	0.8132	0.9008	**0.9809**
Gaussian NB	0.5378	0.7831	**0.8301**	0.5695
QDA	0.5819	0.8040	0.8366	**0.8544**

Table A.4: Training TPR and FPR for each feature category using multiple classifiers

Classifier	URL-based	HTML-based	Web-based	All Features
Logistic Regression	0.5815/0.0918	0.8102/0.2174	0.7687/0.0624	0.8890/0.0711
KNN	0.7943/0.2171	0.8039/0.1566	0.8939/0.0337	0.9998/0.0010
SVM	0.7117/0.1667	0.7990/0.1695	0.8977/0.0513	0.9639/0.0232
Ada Boost	0.8908/0.3499	0.8249/0.2390	0.8682/0.0765	0.9599/0.0399
Decision Tree	0.7739/0.0713	0.8005/0.1618	0.8983/0.0360	0.9998/0.0011
Random Forest	0.8194/0.0897	0.7976/0.1661	0.9373/0.0524	0.9759/0.0073
Gaussian Process	0.7276/0.1685	0.8098/0.1795	0.8666/0.0564	0.9731/0.0108
Gaussian NB	0.3832/0.0429	0.7854/0.2206	0.7812/0.1051	0.4079/0.0239
QDA	0.4284/0.0432	0.8136/0.2165	0.7770/0.0771	0.7620/0.0220

* Format of the data follows TPR/FPR.