TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Papuna Abesadze 184025IVSB

Statistical Learning Methods for Malicious URL Detection

Bachelor's Thesis

Supervisor: Toomas Lepikult
PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Papuna Abesadze 184025IVSB

STATISTILISED ÕPIMEETODID KURITAHTLIKE URL-IDE TUVASTAMISEL

Bakalaureusetöö

Juhendaja: Toomas Lepikult
PhD

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Papuna Abesadze

24.03.2022

Abstract

The contents of this paper are based on the data gathered during the tests conducted by the author. It examines all the major methods how one could approach the challenge of detecting malicious URLs based solely on the components of the URL by formulating it as a binary classification problem. The paper offers a comprehensive comparison of 3 predominant predictor variable classes (Lexical, Linguistic, Host-Based) and assesses their efficacy for predictive modelling, in conjunction with their complexity and challenges of computability, while evaluating each class of predictors on 10 learning algorithms. The paper aims to ascertain which classes of predictors are best suited for predictive modelling in different circumstances, and which learning algorithms tend to perform best for each evaluated set of predictors.

The analysis concludes by providing valuable information for future researchers about expected challenges, potential complications, and analytical techniques to be aware of should they decide to adopt and further research predictive models examined in this paper.

This thesis is written in English and is 46 pages long, including 6 chapters, 11 figures and 5 tables.

Annotatsioon

Statistilised õpimeetodid kuritahtlike URL-ide tuvastamisel

Töö baseerub autori poolt lõputöö käigus läbiviidud testide andmetel. Töös uuritakse peamisi meetodeid, mille abil leitakse pahatahtlikud URL-aadressid. Seejuures kasutatakse üksnes URL-i komponente. Ülesanne formuleeritakse binaarse klassifitseerimise ülesandena. Töös esitatakse kõikehõlmav võrdlus kolme domineeriva prognoosimuutuja klassi (leksikaalne, lingvistiline, võõrustajapõhine) vahel. Hinnatakse nende efektiivsust modelleerimisel koos arvutusliku keerukusega. Hindamisel kasutatakse kümmet õpialgoritmi iga prognoosimuutuja klassi jaoks. Töö eesmärgiks on tuvastada, millise klassi prognoosimuutujad sobivad kõige paremini erinevates olukordades. Samuti uuritakse, milline õpialgoritm osutub efektiivseimaks iga prognoosimuutuja klassi puhul.

Analüüsi tulemusena esitatakse oluline teave tulevastele uurijatele probleemi väljakutsete, potentsiaalsete probleemsete kohtade ja andalüüsitehnikate kohta, mida tuleks prognoosimudelite edasisel rakendamisael ja täiustamisel arvesse võtta.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 46 leheküljel, 6 peatükki, 11 joonist, 5 tabelit.

List of abbreviations and terms

CV	Computer Vision
NLP	Natural Language Processing
ML	Machine Learning
URL	Uniform Resource Locator
SVC	Support Vector Classifier
LR	Logistic Regression
DSP	Digital Signal Processing
TTL	Time-To-Live
DNS	Domain Name System
ISP	Internet Service Provider
KNN	K-Nearest Neighbours
LR	Logistic Regression
NB	Naïve Bayes
RF	Random Forest
DT	Decision Tree
BC	Bagging Classifier
SGDC	Stochastic Gradient Descent Classifier
DNN	Artificial Deep Neural Network
GBC	Gradient Boost Classifier

Table of contents

List of figures

List of tables

1 Introduction

Chapter 1.1 outlines legacy methods for detecting malicious URLs, chapter 1.2 provides background information about the problem this research will aim to resolve, followed by a formal definition of assignment in the chapter 1.3, assumed limitations in the chapter 1.4 and information about the structure of thesis in the chapter 1.5.

1.1 Non-Statistical Methods of Pinpointing Malicious URLs

Historically the problem of detecting malicious URLs (Uniform Resource Locators) was combatted against by compiling and maintaining databases of known harmful URLs. However, the so called "Reputation Based Blacklisting" technique did not prove to be sustainable, for as the number of records kept increasing, so did the computational complexity of performing database lookups. Moreover, in 2008 [1] have demonstrated that no such database can be perfectly, or even sufficiently inclusive and more importantly, precise. More malicious hosts emerged per unit time than could be Blacklisted, and even in case this shortcoming could somehow be overcome, the nature of type I and type II decision errors during network and host analysis of potentially harmful URLs were nontrivial. Thus, a need arose of finding ways to perform malicious URL detection proactively rather than retroactively. One of the first ways of doing real time analysis was rule-based approach. It entailed dynamical analysis of website behaviour such as frequency of redirections, and its contents, particularly JavaScript. It was certainly more accurate and scalable, however, as [2] argues, the need for loading website content before making the decision puts the end users at the risk of triggering malicious executables – the very threat this solution aimed to mitigate, thus defying its purpose.

In 2011 an influential paper "Learning to Detect Malicious URLs" came out [3] which drew upon most of the relevant foundational research that had already been done by that time, great part of which focused merely on feature engineering, and attempted to implement a real-time Malicious URL predictor using nothing but predictor variables extracted from URLs and a learning algorithm, moving away from webpage behaviour and content analysis.

1.2 Description of the problem

We will come back to the chronology of developments in this field and overview related work in the consequent chapters, but the central point is that in the recent years several serviceable papers have been published which attempted to propose new predictor variables and learning algorithms that were reportedly more efficacious than previous approaches. However, despite all the effort, to this day, unlike in the fields of Computer Vision (CV) and Natural Language Processing (NLP) wherein certain solutions are conspicuously superior to others, such as Convolutional Neural Networks for CV and Transformers for NLP, and where we already have pre-trained models such as BERT (NLP) and ResNet (CV), for the problem of detecting Malicious URLs it remains largely unclear how the efficacy of different proposed solutions compares objectively. One of the contributing factors to this persisting uncertainly is the problem of reproducibility. Claiming that one's Machine Learning model achieved, suppose, 95% accuracy compared to 80% claimed in a certain other paper is not an informative and statistically accurate piece of information due to the following reasons:

1. The same learning algorithm evaluated on different data sets will almost certainly yield different results. Normally, if the size of the training dataset was sufficiently large, model performance should not vary substantially. However, sampling logic matters greatly. It is crucial for objective evaluation that the sample is representative. For otherwise, If the sample is biased, this could lead to the exclusion of certain types of URLs on which the model would have performed poorly, leading to the artificial increase in the model's accuracy while, in reality, the model is not as accurate as performance test results would lead one to believe.

2. Same learning algorithm will perform differently even on the same dataset if the feature engineering was handled differently. Thus, paper A claiming that their Support Vector Classifier trained on a feature set $S_1 = [a_1, a_2, a_{3...}a_n]$ which had achieved 95% accuracy is superior approach to paper B's Logistic Regression

with the accuracy score of 90%, trained on the feature set $S_2 = [b_1, b_2, b_3 ... b_n]$, while in part true, is misleading, for we do not know how these learning algorithms would have performed and compared to each other if they had been trained on the same set of features. Same applies to comparing approaches to feature engineering.

This is a recurring theme the reader will observe in the section dedicated to related work. While many excellent pieces of research have been published with the aim of developing techniques for identifying malicious URLs using Machine Learning, either the datasets on which their models were trained, or data preparation and feature extraction pipelines they chose, or all the above vary substantially across many pieces of research, making it challenging to perform cross-comparisons to ascertain:

1. which approaches to feature extraction produce the best predictors and
2. given a preferred feature extraction method, which statistical learning algorithms perform most efficaciously.

Furthermore, most published researches merely cherry-pick 2 or 3 algorithms that score best given their set of assumptions and a choice of data preprocessing pipelines while not disclosing methodology they used for sampling training and testing data. There has not yet been a comprehensive evaluation and comparison of all major proposed approaches performed on equal grounds – the shortfall this paper strives to address.

1.3 Formulation of the assignment

The aim of this paper is to replicate and conduct comparisons between all three major proposed methods for detecting malicious URLs solely based on the anatomy of the URL itself on the dataset of over 650,000 observations, and offer thorough analysis of obtained results to ascertain:

- Which approach to feature extraction yields variables with the highest predictive power: Linguistic Features (Applying natural language processing techniques to URLs), Lexical Features (derived mathematical features calculated from various

URL components), or Host-based Features (performing host-related lookups in public databases, such as WHOIS).

- Which learning algorithms with which hyperparameters perform best given a preferred set of predictor variables. 10 learning algorithms will be trained and evaluated on each set of features.

- Lastly, this paper provides a comprehensive evaluation of advantages and potential complications associated with extracting and employing each set of features for predictive modelling, including complexity of feature engineering, computational time, and essential system resources usage. Moreover, the analysis concludes by providing anyone interested in pursuing this research with valuable information such as which classes of learning algorithms would be optimal to use on different sets of features, what are some of the problems associated with retrieving host-related data from public databases, which approaches to predictive modeling do and do not work and more. The analysis, thus, strives to aid future researchers navigate the vast landscape of possibilities by offering the ultimate guide to which methodologies are worth exploring assuming their research aims and resources.

1.4 Limitations

Due to the scale of the conducted experiment and time restraints, certain restrictions had to be imposed to narrow down the research problem.

Firstly, due to the nature of the labeled dataset and the fact that dependent variable only contains 2 classes (Malicious/Benign), detecting Malicious URLs will be approached as a supervised learning, more specifically, as a binary classification problem. Unsupervised learning algorithms, therefore, will not be evaluated. This also implies that the aim of predictive models is not to ascertain exact ways in which given URLs are malicious, but merely to make a binary prediction – Malicious or Not Malicious. Topics in web application security such as spam, phishing, malware, and defacement URLs will, therefore, not be covered.

1.5 Division of chapters

Chapter 2 provides essential information about the anatomy of URLs, and how their different components can be engineered to act as predictors for maliciousness in 3 different ways. Chapter 2.3 offers a big-picture summary of related work in this research field and describes shortcomings in their methods. Chapter 3 expounds methodology and technical details behind the conducted experiment, followed by result analysis in chapter 4, future research recommendations in chapter 5 and a final summary in chapter 6.

2 Background

Chapter 2.1 outlines components of URLs which are particularly efficacious for generating predictors. Chapter 2.2 explains the details behind generating each class of predictors, followed by an overview of related work on this issue in the chapter 2.3.

2.1 The Anatomy of the URL

URLs, unlike human languages, do not possess grammatical or inherent semantic properties. However, their formatting still follows a strict syntax (see Figure 1).

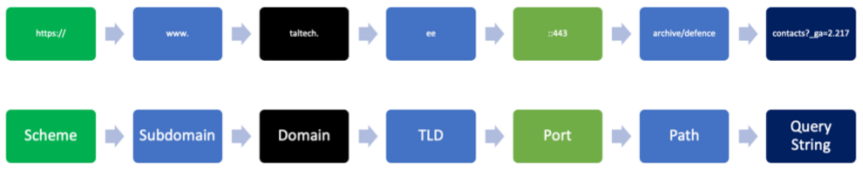


Figure 1. Components of the URL.

Due to this fact, URLs can be decomposed into different subcomponents, some of which are more helpful for identifying a potential security threat than others. Here are some of the URL components that are generally agreed to be statistically relevant for building predictive models [4]:

- Scheme – specifies which protocol to use for retrieving a specific resource. In the dataset on which predicted models were trained, schemes were mostly http:// or https:// and did not vary substantially amongst malicious and benign data points. But URL scheme need not be limited to the aforementioned two, scheme may also be ftp://, mailto:// and others.

- Hostname - can be further broken down into subdomain, domain and top-level domain as shown in the Figure 1. Hostname, particularly if it contains IP address, can be highly useful for building predictive models.

- Path – specifies directory route and a specific resource in the destination which needs to be accessed. Frequently used for generating lexical features. (See chapter 2.2)

- Query String – is not always present, but it is usually a string of key-value pairs, providing an information that the resource can use for some purpose [4]. Query strings are heavily used for generating predictor variables using Natural Language Processing techniques.

2.2 Generating Predictors

In machine learning terms the process of generating predictor variables is called Feature Engineering which is how it will be referred to from here onwards. Generally, any numerical or categorical variable whose values are correlated with the values of the dependent variable (in our case, whether a given URL is malicious), is considered to be a feature.

Examples of features include matrices of bytes with each byte corresponding to a pixel in the image for the problems of Computer Vision, word embeddings (vectorized words) for NLP, and variables such as frequency, phase, and amplitude for Digital Signal Processing. However, for detecting Malicious URLs feature extraction is neither as intuitive nor so straightforward since URLs do not have a one-to-one numeric representation unlike images and digital signals.

[5] and [6] were some of the first papers which examined how malicious URLs tend to vary from benign ones, what are the parameters presence of which might help detect any potential threat, and how to use them for model building. Broadly speaking, if we are to leave the webpage content information aside and perform URL evaluation solely based on its components, predictor features can be divided into three main categories:

1. Lexical Features – usually numerical variables calculated based on statistical properties of URLs, such as the number of ‘\’, ‘@’, ‘.’, '?’, ‘=’, ‘-‘, etc. More sophisticated Lexical features include special-to-normal character ratio, a Boolean indicating the use of shortening services, subdirectory count in the path, number of subdomains, relative length of the URL considering interquartile range, and measures calculated using information theory such as text entropy.

2. Host Features – mostly categorical variables obtained as a result of performing host-related lookups in the public databases. WHOIS queries are one example. [5] have argued that specific pieces of information about given host, such as date of registration and expiration, host registrar and registrant, registration country, TTL for the corresponding DNS record and connection bandwidth can be effectively used to quantify the likelihood of the URL associated with this host being malicious.

3. Linguistic Features – sometimes also known as tokenized features, or “Bag-of-Words” method. There is no strict definition of what criteria a feature must satisfy to fall under this category, however, normally, any set of predictor variables obtained by employing the techniques initially developed for NLP is considered Linguistic. The end result of generating Linguistic Features is a set of abstract, vectorised URL components which can be clustered together using some distance metric d(x, y…n) defined on the obtained n-dimensional vector space to ascertain which feature vectors “look” more similar to malicious URL components and which vectors resemble benign URL tokens.

During the conducted analysis 25 lexical and 15 most commonly used host features were selected as an input for learning algorithms. A detailed list of employed predictors is given in the tables 1 and 2.

2.2.1 Lexical Features

Table 1 provides a detailed explanation of selected Lexical features and how they are derived.

Table 1. Selected Lexical Features.

Feature	Description
url_contains_ip	Boolean. True IFF at least one substring of the URL is a valid IPv4 or IPv6 address.
url_length	Count of all characters in the URL string.
path_length	Count of all characters in the URL path.

Feature	Description
host_length	Count of all characters in the URL host.
contains_port	Boolean. True IFF domain is followed by a port number.
digit_count	Count of numeric characters in the URL string.
query_parameter_count	Count of query parameters in the URL's query string. (See Figure 1)
query_fragment_count	Count of query fragments in the URL's query string. Fragments usually reference elements with specific IDs in the document and are preceded with #.
encoded	Boolean. True IFF URL displays signs of being encoded.
encoded_char_count	Count of encoded characters in the URL string.
subdir_count	Count of subdirectories in URL's path.
url_string_entropy	Shannon entropy of the entire URL string. (See Equation 1).
dot_count	Count of dots in the URL string.
client_substring_present	Boolean. True IFF at least one of the substrings of the URL is "client".
admin_substring_present	Boolean. True IFF at least one of the substrings of the URL is "admin".

Feature	Description
server_substring_present	Boolean. True IFF at least one of the substrings of the URL is “server”.
login_substring_present	Boolean. True IFF at least one of the substrings of the URL is “login”.
norm_char_count	Count of normal English characters (a-z) in the URL string.
digit_count	Count of Arabic numerals (0-9) in the URL string.
is_shortened	Boolean. True IFF pattern matching identifies the use of at least one of top free shortening services.
special_char_count	Count of special characters in the URL string.
spec_ratio	Ratio of special the special character count and the length of the entire URL.
norm_ratio	Ratio of normal character count and the length of the entire URL.
numeric_ratio	Ratio of count of digits and the length of the entire URL.
url_length_agg	Aggregated URL length taking into account interquartile range of all URL lengths in the dataset. Value Range [1…4]

The only feature that needs further clarification is “string_text_entropy”. Shannon Entropy is a measure first introduced in information theory by Claude Elwood Shannon.

It is directly correlated with quantifying how much information a given set of characters contains. If we assume an argument to be a string of characters, as it was for the dataset of malicious URLs, text entropy can be calculated using the following equation:

$$H(x) = -\sum_{x} P(x) \log_2 P(x) \qquad (1)$$

Where P(x), for each character in the string, is count of occurrences of that particular character divided by the length of the entire string. [7, pp. 73-74]

2.2.2 Host Features

Table 2 outlines Host Features selected for predictive modelling, and how each of them are constructed.

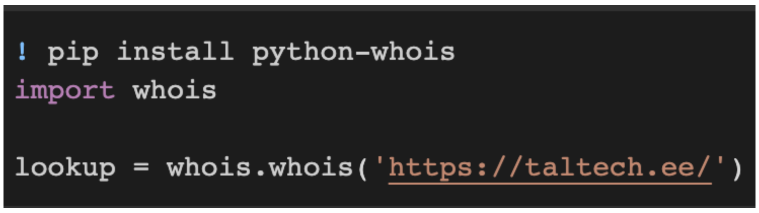
Table 2. Selected Host Features.

Feature	Description
subdomain_count	Count of distinct subdomains that are registered for the host of a given URL.
url_age	CURRENT_TIMESTAMP – (domain registration date according to WHOIS records)
url_intended_life_span	The difference between expiration and registration dates according to WHOIS.
url_life_remaining	(Domain expiration date according to WHOIS) - CURRENT_TIMESTAMP
url_registrar	Registrar of host according to WHOIS.
url_registration_country	WHOIS lookup.
url_host_country	WHOIS lookup.
open_ports_count	Count of distinct open ports.

os	Operating System of the platform where the webpage is hosted.
days_since_last_seen	(Count of days since the webpage was last seen online according to snapshots) - CURRENT_TIMESTAMP
days_since_first_seen	(Count of days since the webpage was first seen online according to snapshots) - CURRENT_TIMESTAMP
mean_update_frequency	Mean number of days it takes for the webpage to update according to its snapshots.
number_of_updates	Total number of times a given webpage was updated, according to snapshots.
ttl_from_registration	TTL for the corresponding DNS record
isp	Internet service provider of the platform where a given webpage is hosted.

Unlike Lexical and Linguistic features, deriving Host Features based on URL is not a self-contained procedure. It involves making a multitude of API calls to query public databases and, consequently, additional layers of pre-processing. As [5] argue, one of the lookups which can provide valuable statistical information about host is WHOIS lookup. The reasoning behind this suggestion is that every time a new domain is registered, contact and identification information must be provided such as name, mailing address, Email, phone number, city, postal code, and country [8] which, due to its privacy policy, can later be retrieved by anyone using WHOIS lookup [9].

Using python API, performing such lookup can be done by passing the URL as an argument, as depicted in the Figure 2:

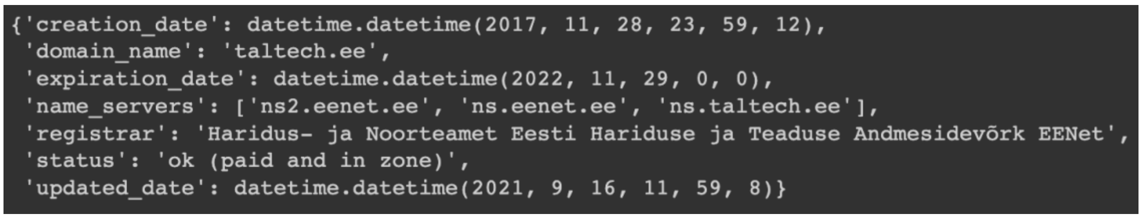


```
! pip install python-whois
import whois

lookup = whois.whois('https://taltech.ee/')
```

Figure 2. Performing WHOIS lookup.

The returned is the “whois.parser” object containing semi structured data, ready for pre-processing:



```
{'creation_date': datetime.datetime(2017, 11, 28, 23, 59, 12),
 'domain_name': 'taltech.ee',
 'expiration_date': datetime.datetime(2022, 11, 29, 0, 0),
 'name_servers': ['ns2.eenet.ee', 'ns.eenet.ee', 'ns.taltech.ee'],
 'registrar': 'Haridus- ja Noorteamet Eesti Hariduse ja Teaduse Andmesidevõrk EENet',
 'status': 'ok (paid and in zone)',
 'updated_date': datetime.datetime(2021, 9, 16, 11, 59, 8)}
```

Figure 3. Returned WHOIS object.

From the Figure 3 it is apparent that, while valuable, data returned by the WHOIS lookup is limited and does not allow calculation of more elaborate features listed in the Table 2. [23] have demonstrated how to overcome these limitations and retrieve more sophisticated host data proposed by [5] and [6] using Python’s Shodan and Wayback Machine libraires. Their approach was partially reused as a starting point for this stage of feature engineering.

2.2.3 Linguistic Features

Generating linguistic features is computationally much more straightforward and self-contained procedure. The first stage is decomposing URLs into their constituent parts as depicted in the Figure 1. This is often called tokenization. The second stage involves filtering out tokens whose distributions do not change significantly between the two classes (Malicious and Benign) to reduce the dimensionality of the resultant vector space. Such tokens are called “stop words” – words that appear far too frequently to be statistically significant for predictive modelling. These can include different parts of speech such as pronouns. In case of URLs, potential stop words are URL scheme, “www”, “.com” etc. [11, p. 70].

The third and final stage of linguistic feature generation entails addressing representation problem – the process of converting obtained URL tokens to numeric values that can be parsed by learning algorithms. Most papers investigating Malicious URLs default to using “Bag of Words” approach. It involves vectorizing obtained tokens based on their occurrence frequencies. While more sophisticated N-Gram vectorization methods have been proposed in NLP which allow “meaning” of tokens to be captured more precisely [12, p. 203], URLs do not have semantic structure unlike human languages, hence, during conducted analysis conventional unigram “Bag of Words” method was employed.

A detailed performance comparison of these 3 sets of features, in conjunction with the review of challenges and potential complications associated with using them for predictive modelling will be done in the chapter 4.5. For the readers who would like to find out more technical information about everything discussed in this paper, data pre-processing script in its entirety, along with predictive modelling script and all collected datasets is available at [10].

2.3 Related Work

Most of the relevant related work has already been mentioned in the chapters dedicated to presenting historical perspective, formulating the research problem, and describing employed feature engineering methods. This chapter draws the reader’s attention to supplementary pieces of research that would be valuable resource for anyone investigating malicious URL detection techniques and highlights additional ways in which this paper differs from them.

Broadly speaking, most papers that have been published in this field can be divided in the categories depicted in the Figure 4:

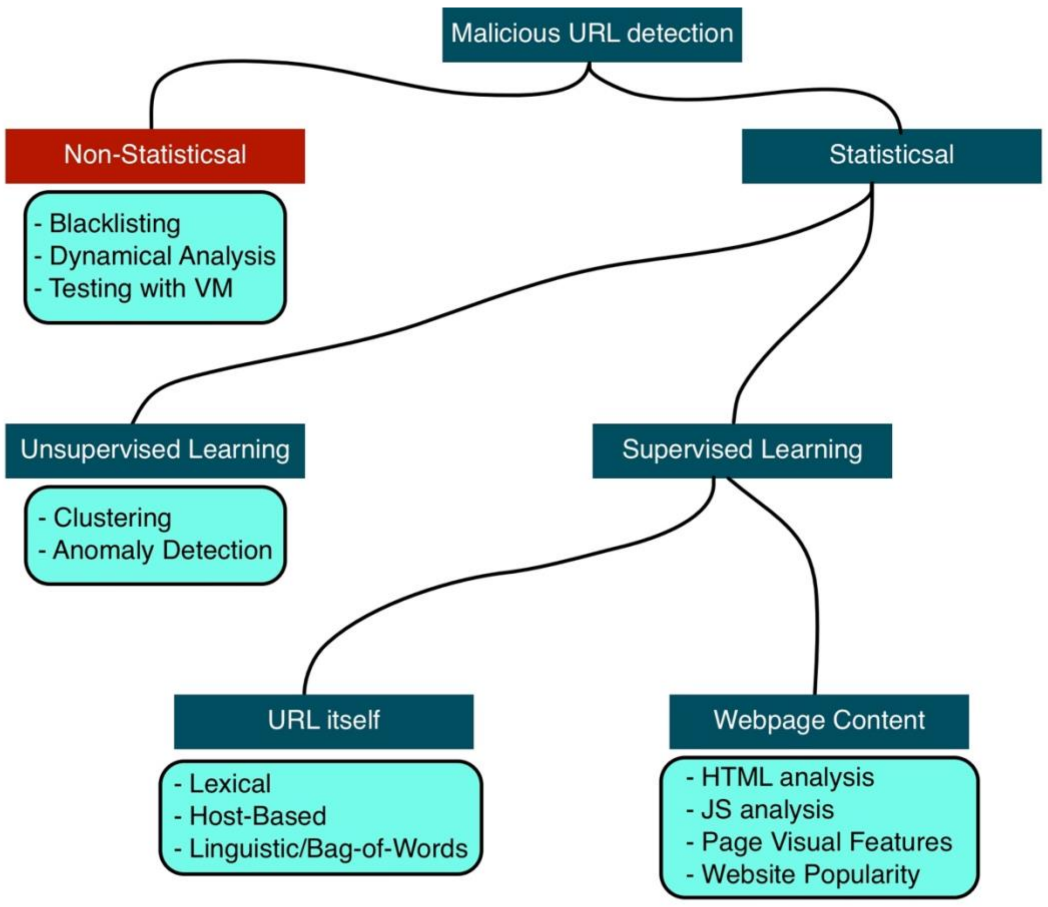


Figure 4. Techniques of detecting Malicious URLs.

Information illustrated in the Figure 4 was based on the data gathered in the recent survey [13] which is a comprehensive technical summary of over 150 papers published in this field. It is apparent from the Figure 4 that this paper is only concerned with the approach which is but one of vast number of possible solutions proposed in the recent years. Moreover, while other papers focus too narrowly on specific business problems, and investigate challenges associated with deploying obtained predictive models to production, this paper focuses on comparing and thoroughly evaluating all the different proposed methods of assembling predictive models themselves, without deployment constrains, by testing 10 learning algorithms on 3 different sets of features. The only other paper that comes close in the number of evaluated learning algorithms is [14]. However, they do not cover first order optimisation methods such as Gradient Boosting classification models discussed in this paper, nor do they make any attempts to evaluate them on lexical, host and linguistic features which are thoroughly covered in this paper.

Lastly, this work is a comprehensive evaluation of all major URL-based predictive methods propounded from the early 2000s onwards, tested on equal grounds, which is analysis that has not been conducted before. The contents of this paper will aid future researchers navigate the vast landscape of possibilities by providing valuable information about advantages, shortcomings, challenges, and potential complications associated with adopting each of the methods evaluated in this research.

3 Methodology and Tooling

The research that lies at the foundation of this paper involved experimentation stage followed by the theoretical analysis of obtained empirical data. This section provides technical information about conducted experiment, assumptions made and employed tooling.

At the centre of conducted experiment was the dataset of over 650,000 Malicious and Benign URLs with the following dependent variable distribution (Figure 5):

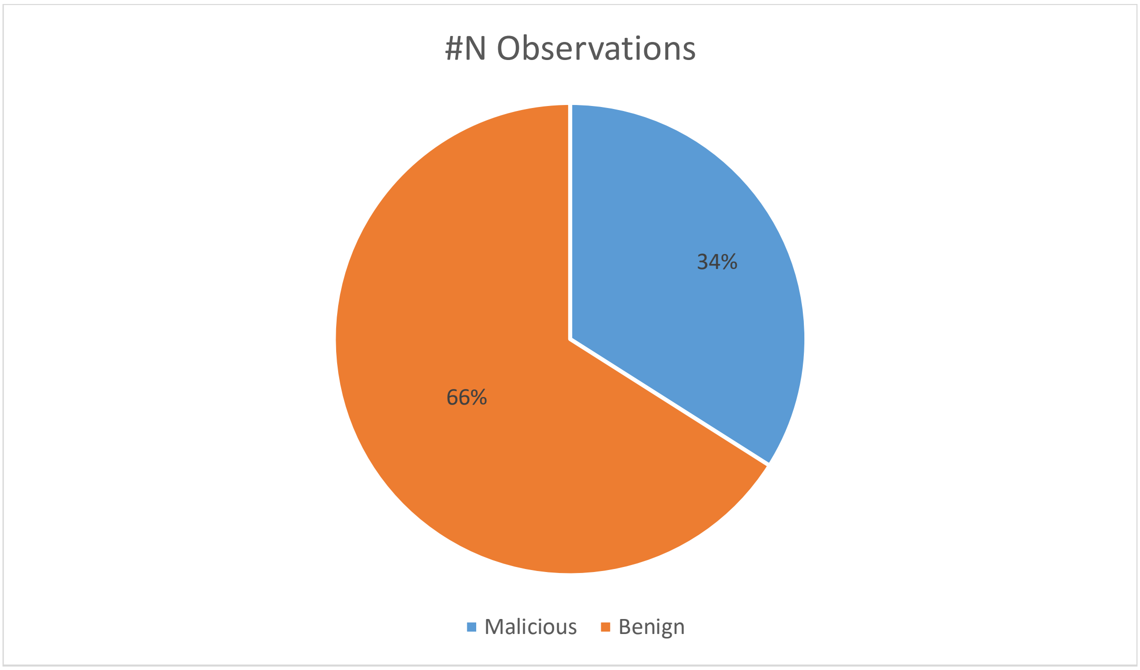


Figure 5. Class imbalance in the dataset.

Since the dataset is labelled and dependent variable only has 2 possible classes, this makes malicious URL detection a supervised learning, more particularly, a binary classification problem.

During the course of conducted analysis 2 Python3 scripts were developed. Using predominantly Pandas and NumPy libraires, the first script parses and cleans the dataset, performs foundational pre-processing, and handles feature engineering for lexical and host features. The second script takes derived datasets as an input and handles predictive modelling stage. Using predominantly Python's Scikit-Learn and Tensorflow2 libraires, it implements 10 learning algorithms and thoroughly evaluates them on all 3 sets of features. Chosen algorithms, as well as metrics used for their evaluation are discussed in detail in the Chapter 4. Both scripts are available for viewing at [10]. Code samples are provided in the Jupyter Notebook format and were executed on Google Cloud's hosted runtimes [15] for security reasons. Computational complexity of the 3 sets of features and evaluated learning algorithms is discussed in the chapter 4.

4 Research and Analysis

Chapter 4.1 formally defines the process of detecting Malicious URLs as a binary classification problem and introduces learning algorithms used during conducted analysis. Chapter 4.2 describes statistical methods used during data preparation. Chapter 4.3 formulates metrics using which the efficacy of learning algorithms was evaluated followed by chapters 4.4 - 4.5 which outline obtained results in detail.

4.1 Learning Algorithms

As mentioned in the Chapter 3, due to the nature of the dataset, the process of detecting malicious URLs can be assumed to be a binary classification problem. One can think of any binary classification algorithm as some multivariable function $g(x_1 \ldots x_n)$ which accepts n number of predictor variables as arguments and outputs either of the 2 possible values for the dependent variable Y.

Prediction Class $Y = g(X_1 \ldots X_n)$; $n \in \mathbb{N}$. (2)

During the conducted experiment $n_{\text{Lexical}} = 25$, $n_{\text{Host}} = 15$ and $Y \in ['Malicious', Benign']$. Input dimensionality of linguistic features is slightly more nuanced and will be explained in the later chapters.

Equation (2) is manifestly a broad generalisation of binary classification logic. Each learning algorithm tends to be different in terms of which mathematical methods they use to perform the mapping of each set of independent variables $X_1 \ldots X_n$ to the dependent variable Y. The algorithms that were chosen for predictive modelling are the following:

1. K-Nearest Neighbours (KNN) – “K” in the “KNN” is a free parameter. It is an integer indicating how many neighbouring observations an algorithm should use to perform classification. KNN embeds each observation, in our case URL data in a feature space and deicides which class a given URL belongs to according to the majority class of k-number of closest observations to it.

2. Logistic Regression (LR) – Despite its name, LR is mostly used for classification problems. Using logistic function, it calculates the probability of a given observation (URL) belonging to a particular class (Malicious/Benign), given a set of corresponding predictor variables $X_1 \ldots X_n$. By default, if the probability exceeds 0.5, observation is classified as Malicious. However, decision boundary can be altered.

3. Naïve Bayes (NB) – At the heart of Naïve Bayes Classifier lies the Bayes Theorem for conditional probability. NB tends to achieve performance comparable to that of linear classifiers such as LR and SVC, however its computational complexity is usually lower as it learns parameters by analysing individual predictor variables to infer basic class-wise statistics for each independent variable [16, pp 68].

4. Support Vector Classifier (SVC) – The key idea behind SVC is the fact that it tries to separate observations to different classes as effectively as possible. To do so, SVC attempts to come up with what is called a decision boundary and optimise it using a distance metric. Since observations are represented as points in n-dimensional space, in 2D decision boundary is a “separation line” between these points, in higher dimensions it becomes a hyperplane [17 pp 145-148]. Observations that lie closest to the decision boundary are called support vectors, hence the name.

5. Decision Tree (DT) – is a tree-based classification technique which excels in cases when the dataset is relatively complex, and observations are not linearly separable. That is to say, one cannot come up with a line, or hyperplane equation which accurately divides observations in distinct classes [17, chapter 6]. Shortcomings of using DT include model interpretability as it can sometimes be

difficult to make sense of how the decisions are made, and its predisposition to overfitting (see Chapter 4.3).

6. Random Forest (RF) – is similar to simplistic DT, however RF is what is called an ensemble method, meaning that it makes use of multiple instances of the learning algorithm simultaneously for decision-making, and hence eliminates the overfitting problem. Number of Decision Tree instances used for Random Forest is a free parameter that can be specified. But generally, if one chooses K instances, the dataset will be randomly sampled into train-test sets K number of times, with each time new instance of Decision Tree being trained on the training set. Once this stage is complete, we are left with K number of Decision Trees trained on different datasets due to random sampling. During prediction, all K number of DTs will be asked to predict the class of a given observation. Final answer is mode of resultant K predictions [18 pp 320-321].

7. Bagging Classifier (BC) – Bagging is the name of the method used by RF to repeatedly re-sample the data and train multiple instances of the learning algorithm on them. Unlike RF though, BC does not randomly subset a set of predictor variables used by each instance of the learning algorithm.

8. Gradient Boosting Classifier (GBC) – Boosting is an ensemble method much like Bagging, designed to improve upon simplistic Decision Trees. Similarly to Bagging, it entails implementing multiple instances of the learning algorithm, but what's different is that Boosting builds DTs in a sequential manner, each learning and improving upon previous models. "Improvements" are quantified by tracking incremental minimization of a chosen a loss function. [16, pp 88 - 92], [18, pp 321 - 324].

9. Stochastic Gradient Descent Classifier (SGDC) – Gradient Descent and Stochastic Gradient Descent are not, on their own, classes of learning algorithms. Rather, they are what are called optimization algorithms – mathematical techniques for minimizing a given loss function and, consequently, maximizing the accuracy of models [17, pp 111-119]. For instance, one can initiate Logistic Regression with the gradient descent learning technique instead of the default solver [19].

10. Artificial Deep Neural Network (DNN) – Much like the previous example, DNNs use Gradient Descent as a method for minimizing the loss function. During conducted testing, two 2-layer DNNs were implemented for lexical and host features using Keras sequential API. The input layers had the same dimensionality as corresponding feature vectors, with activation function set to relu. Output layers contained a single node with sigmoid activation function. Both networks were compiled using Adam optimizer with the binary cross-entropy selected as a loss function. For readers who would like to find out more technical details, Python notebooks can be found at [10].

4.2 Data Preparation

Chapter 4.2.1 elucidates one of the most important characteristics of any dataset – its dimensionality. We will see in the later chapters how overly high dimensional datasets can have a negative influence on learning algorithms. Chapter 4.2.2 outlines essential statistical methods for transforming datasets such that they can be used for predictive modelling.

4.2.1 Input Dimensionality

After running feature engineering script described in the chapter 2.2, we are left with 3 datasets each containing a distinct set of features. However, their dimensionality varies in a nontrivial manner. The shape of the matrix containing Lexical data is 651k x 25 meaning that we have 651,000 observations to work with, each portrayed by 25 independent variables.

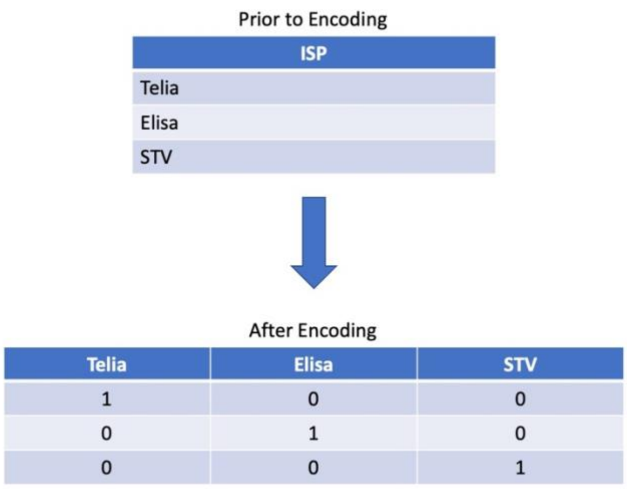
Once we move on to the Host data, dimensionality drops to 188k x 15 – more than 70% reduction in the number of observations. One of the reasons behind this is the quality of the dataset itself. Since host features are generated by the means of performing public database lookups related to the URL host, the lookup will fail if URL parsing library fails to extract host [21]. However, a larger problem is the fact that Malicious websites tend to have a much shorter lifespan than genuine ones [2] and therefore, if host features are not collected in a near real-time, they are likely to be forever lost. This is one of the problems associated with using host features for predictive modelling that we will come back to in the later chapters.

In regard to Linguistic features, the dimensionality of the dataset amounted to 651k x 824k – significantly larger than the other two sets of features. The reason behind this enormous size is how count-vectorisation works. Every distinct URL token is, essentially, represented as a Boolean value in the new column. As we will see in the following chapters, the matrix of this size is likely to cause hardware limitation problems.

4.2.2 Encoding and Rescaling

During predictive model building, it is an essential step to sample datasets into training and testing data, for it makes no sense to evaluate the models on the same data they were trained on. Since the number of observations available to us was on the scale of hundreds of thousands, approximately 75% of data points were allocated to training.

Since it is required by chosen learning algorithms for all data points to be numerical values, the next stage of data preparation is handling categorical features such as host ISP, domain registrar, etc. This can be done by one-hot-encoding depicted in the Figure 6:


Prior to Encoding

ISP
Telia
Elisa
STV

After Encoding

Telia	Elisa	STV
1	0	0
0	1	0
0	0	1

Figure 6. Encoding Categorical Feature.

Note that after performing one-hot-encoding, the 2nd dimension of the data frame will increase by the same amount as the count of distinct entries in each categorical column summed over all categorical columns.

The final stage of data preparation is addressing varying scales, or "units" among different features. Learning algorithms that make use of Euclidean distance metric or gradient descent are sensitive to these variations [20, pp 194]. It is, therefore, crucial to rescale data such that the difference between the mean values of any two columns is no longer too large. During data preparation this was done using StandardScaler which subtracts column mean from each observation and divides the result by standard deviation [22].

4.3 Performance Metrics

In order to quantify the efficacy of predictive models, one has to introduce metrics with which to measure the performance first. Since the assignment at hand is to accurately predict whether a given URL is malicious, it would make sense to begin with the plain Accuracy.

4.3.1 Confusion Matrix and Accuracy

Since binary classifiers, by definition, can only produce 4 types of predictions: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), one can assemble these statistics in a mathematical object called Confusion Matrix which is a handy tool for calculating more sophisticated measures: (see Figure 7).

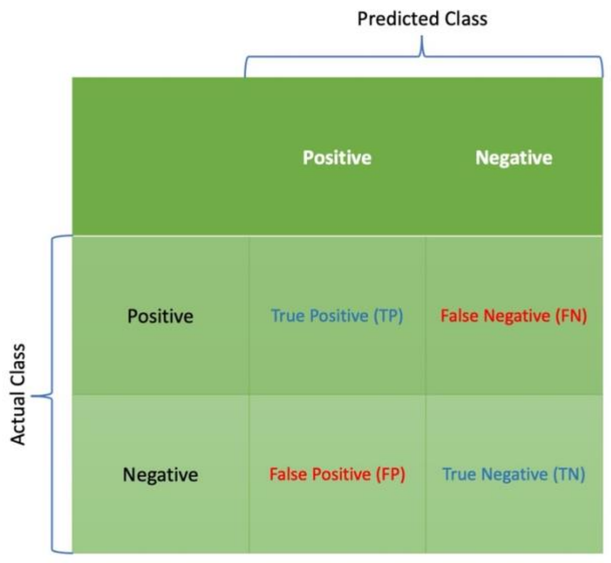


Figure 7. Confusion Matrix of Binary Classifier.

Accuracy, then, would be a ratio of correct decisions and all decisions:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

4.3.2 Recall (True Positive Rate)

It should be noted that plain accuracy is a dataset dependent metric and is not informative in cases where class imbalance is present. Consider the scenario where the predictive model has been deployed to production where Malicious URLs only correspond to a mere 1% of the traffic. Even if predictor were to utterly fail to detect any Malicious URLs and classify all samples as benign, prediction accuracy would still amount to 99% which is clearly deceiving. To counter this problem, one can define the true positive rate of classifier as follows:

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

Recall tells us out of total number of Malicious URLs (TP + FN), what proportion the classifier has managed to classify correctly as Malicious.

4.3.3 Precision (Positive Predictive Value)

Ideally, it would be preferred for the resultant classifier not to mark too many benign URLs as malicious, for this reason it is necessary to introduce positive predictive value:

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

Precision tells us out of total number of URLs that were marked as Malicious by classifier, how many were actually Malicious. To summarize, Precision is oriented on decreasing the number of Benign URLs incorrectly classified as Malicious, while Recall focuses on maximizing the number of correctly detected Malicious URLs.

4.3.4 F-Measure

Since for the problem of Malicious URL detection both, Precision and Recall are relevant, there is a need to introduce a more sophisticated metric, the one that takes both (4) and (5) into consideration:

$$F_\beta = \frac{(1+\beta^2)(Precision\ x\ Recall)}{(\beta^2 \text{ x } Precision) + Recall} \tag{6}$$

F-Beta is essentially a weighted harmonic mean of Precision and Recall where $\{ \beta \in \mathbb{R}^{+} \mid \beta \geq 0 \}$ is a free parameter ascertaining weights [23]. Since for the current assignment both, precision and recall are equally relevant, we can take (6) with $\beta = 1$ to obtain what is called an F_1 score (7):

$$F_1 = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} \tag{7}$$

4.4 Benchmarking Results

All 10 classifiers introduced in the chapter 4.1 were implemented and trained on all 3 sets of features after which each learning algorithm was thoroughly evaluated using metrics introduced in the chapter 4.3. Gathered model performance data is summarized in the tables 3, 4 and 5:

Table 3. Classifier Performance on Lexical Feature Set.

Classifier	Accuracy	Precision	Recall	F_1
KNN	94.231%	94%	94%	94%
LR	86.495%	87%	86%	86%
SVC	92.592%	93%	93%	92%
NB	78.512%	82%	79%	76%
DT	93.498%	93%	93%	93%
RF	94.286%	94%	94%	94%
BC	94.276%	94%	94%	94%
GBC	93.258%	93%	93%	93%
SGDC	86.259%	87%	86%	86%
DNN	93.63%	N/A	N/A	N/A

Table 4. Classifier Performance on Host Feature Set.

Classifier	Accuracy	Precision	Recall	F_1
KNN	98.193%	98%	98%	98%
LR	93.136%	93%	93%	93%
SVC	94.823%	95%	95%	95%
NB	42.389%	86%	42%	44%
DT	98.321%	98%	98%	98%
RF	98.342%	98%	98%	98%
BC	98.329%	98%	98%	98%
GBC	97.946%	98%	98%	98%
SGDC	94.468%	95%	94%	95%
DNN	98.06%	N/A	N/A	N/A

Table 5. Classifier Performance on Linguistic Feature Set.

Classifier	Accuracy	Precision	Recall	F1
KNN	86.163%	88%	86%	86%
LR	94.936%	95%	95%	95%
SVC	N/A	N/A	N/A	N/A
NB	N/A	N/A	N/A	N/A
DT	95.366%	95%	95%	95%
RF	95.555%	96%	96%	96%

BC	95.171%	95%	95%	95%
GBC	87.716%	88%	88%	87%
SGDC	92.171%	93%	92%	92%

4.5 Result Analysis and Conclusion

This section offers summary of gathered experimental data and thus provides answers to the research questions posed in the chapter 1.3.

4.5.1 Comparing Learning Algorithms

Judging by the overall accuracy metric (Equation 3), for all classes of predictor variables ensemble learning algorithms performed best. In particular, Random Forest Classifier has managed to achieve the highest accuracy scores 94.28% - 98.34% for all 3 sets of features, followed closely by Bagging Classifier 94.27% - 98.32% for Lexical and Host features, and Decision Tree 95.36% for Linguistic features (see figure 8).

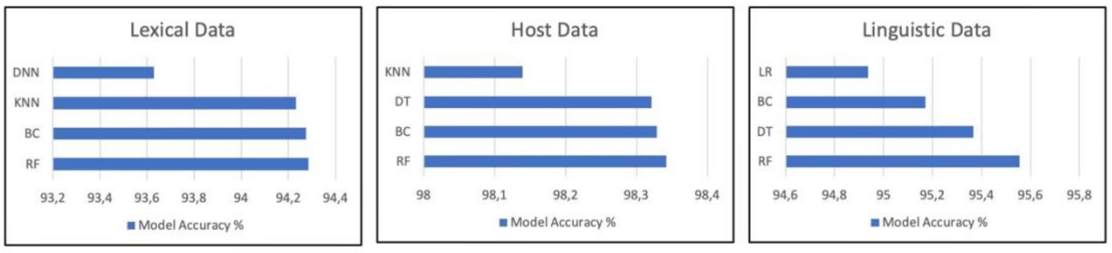


Figure 8. Accuracy Scores of Top Classifiers for Each Feature Set.

It appears continuous resampling and multiple classifier instance approaches employed by ensemble methods such as RF and BC are particularly effective for the nature of the URL-based datasets. It should be noted that linear classifiers such as NB, SVC and LR, which are most frequently used during Malicious URL detection, are not nearly as efficacious as tree-based methods according to the results of the conducted experiment.

Another piece of data that is informative to examine is algorithm training time comparison, which helps ascertain performance to run time ratio, for a certain algorithm may perform 5% more accurately, but it would likely not be worth adopting if its training time is a factor of 10^6 higher.

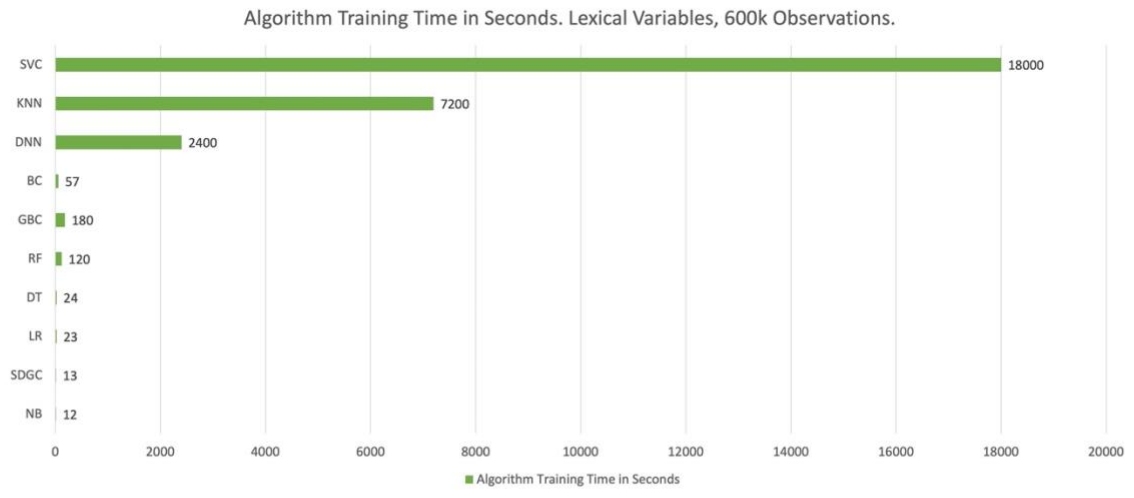


Figure 9. CPU-Training Time of Learning Algorithms.

From the Figure 9 it is apparent that not only were ensemble learning algorithms most accurate, but they were also highly efficient. Furthermore, due to their time complexity, linear classifiers such as SVC and KNN do not scale well on large datasets. Lastly, it should be noted that efficiency of learning algorithms depends on the number of observations they were trained on, DNNs for instance, scale exceptionally well on massive datasets. This will be discussed further in the chapter 5.

4.5.2 Comparing Predictor Variables

Based on data depicted in the figure 8 it is evident that learning algorithms trained on the host variables managed to achieve the highest accuracy score 98.34%, followed by Linguistic variables 95.55% and lastly, Lexical variables 94.28%. However, as was the case for algorithms, assessment of superiority of different variable classes is more nuanced than merely comparing performances of learning algorithms trained on them. One must also take variable computation times and challenges associated with their derivation into account.

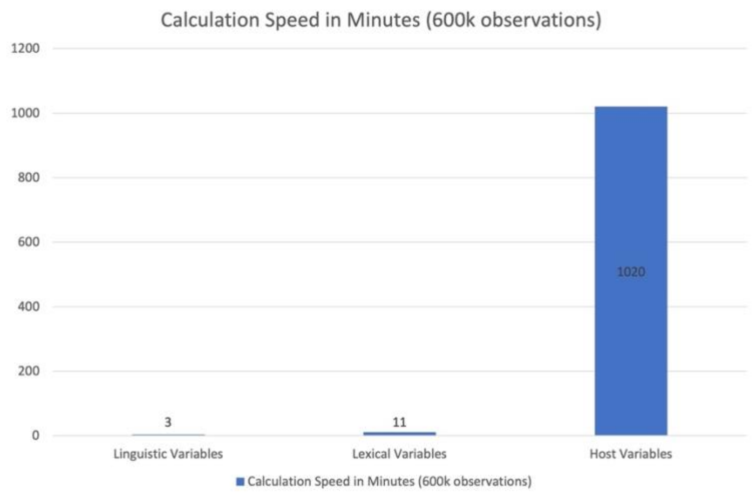


Figure 10. Computation Time of 3 Examined Variable Classes.

As illustrated in the figure 10, computation of Linguistic variables takes the least amount of time since vectorization is a highly optimized operation. Lexical variables take slightly longer - 11 minutes, meaning, this class of predictors still scales well. This is not unexpected since calculation of lexical variables involves basic pattern matching and algebraic manipulations.

Host variables, though, are very different in nature. Due to the need of performing public database lookups, collection of host variables takes nearly 10^3 times longer. What is worse, this is the computation time after optimizing extraction script not to perform duplicated lookups for URLs with the same host. This means, the speed of computation goes up exponentially over time since as the number of performed lookups increases, so does the probability of the next host already being encountered. During the conducted experiment it took 8 hours for the extraction script to calculate host variables for 10% of the data, and additional 9 hours to reach 100%. It is reasonable to assume, therefore, that if one tried to derive host variables the same number of unique hosts, computational time would go up by approximately additional factor of 10. Furthermore, unlike other variable classes for which URL itself is sufficient for computation, host variables require that the webpage to which a given URL corresponds still exists. This is problematic since it has been demonstrated that malicious URLs have a shorter overall lifespan than benign ones [2]. Because of this, only 30% of observations were suitable for computing host variables which caused a 70% reduction in both, available training, and testing data.

To conclude, although learning algorithms trained on host variables perform approximately 4% better, the increase in computational complexity and drop in trainable data quantity makes this class of predictor variables hard to recommend to researchers for adoption.

Based on the conducted analysis it would be reasonable to conclude that lexical variables are overall most optimal for use in the majority of circumstances. While it is true that algorithms trained on linguistic variables manage to achieve a slightly higher accuracy score compared to the ones trained on lexical data, it should also be noted that the dimensionality of linguistic variable data is extraordinarily high, making it impossible for the learning algorithms which do not scale well to the massive amount of features to be trained on them, as depicted in the Table 5 where scores for some algorithms are missing for linguistic variables.

5 Future Research

Due to the nature of thesis work, it was necessary to narrow down the research problem to that which could be accomplishable in the limited time allocated to the corresponding study module. In this chapter outlines how one could extend the scope of the research problem examined in this paper to potentially achieve more precise predictive models, and cover more edge cases, given enough time and resources.

After evaluating almost 30 different combinations of predictor variables and learning algorithms, it became apparent that investing time in feature engineering makes more overall difference than model selection. It is incontrovertible that a superior learning algorithm will almost certainly manage to extract more insights from the data. However, it is more fruitful to try increasing the number of insights itself which can be achieved by increasing the number of effective predictor variables. This can be accomplished by further examining what the visual or statistical differences are amongst benign and malicious URLs, then proposing new derived measures and evaluating their predictive power using correlation tests and chi-squared tests. Not only would this effort help achieve higher classification scores, but it would also expand the scope of covered types of attacks, which in this paper have been limited to defacement, malware, phishing, and spam URLs.

Second method how one could increase the amount of learnable insights is by increasing the dataset size itself. However, unlike in the previous case, being mindful in algorithm selection becomes crucial. One must consider aspects such as computational complexity of algorithms, how well they scale to large datasets, and, most importantly, how the algorithm performance varies as a function of number of observations. A good example of this is depicted in the Figure 11:

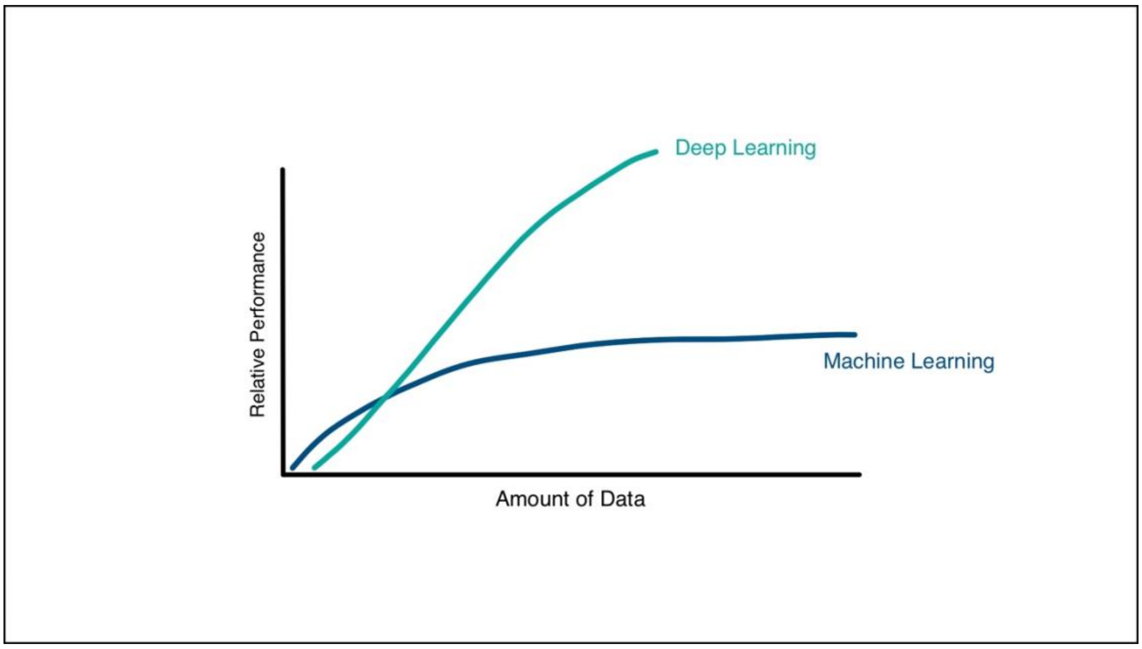


Figure 11. Relative Performance of Learning Algorithms as a Function of Dataset Size. Source: [24].

It is evident from the Figure 11 that while conventional non-deep learning algorithms outmatch Artificial Neural Networks on smaller datasets, once the number of observations grows substantially, Deep Learning manages to extract more insights. It simply scales much better on massive datasets. Furthermore, Deep Learning models can make use of hardware acceleration and, therefore, massive parallelization, which is not an option for most conventional learning algorithms.

Lastly, if collecting high quality annotated data is problematic, it is worth trying to formulate the challenge of detecting malicious URLs as an unsupervised or semi-supervised learning problem. Clustering or anomaly detection could prove to be a fruitful starting point.

6 Summary

The aim of this research was to conduct a comprehensive comparison of 3 different predictor variable classes used for detecting Malicious URLs by evaluating each of them on 10 different learning algorithms to ascertain which classes of predictors were most optimal for use in different circumstances, and which learning algorithms tend to achieve highest accuracy scores for each class of variables.

Relevant findings, in no particular order, are the following:

- Learning algorithms trained on the host variables managed to achieve the highest accuracy score 98.34%, followed by Linguistic variables 95.55% and lastly, Lexical variables 94.28%.
- Lexical variables are most optimal for use in most circumstances due to their short calculation time, high data quality and effectiveness. Linguistic variables are equally quick to calculate, however, dimensionality of this type of data is exceptionally high, making it impossible to be used for training certain types of learning algorithms which do not scale well to high number of features. Linguistic variables would be a good choice for circumstances where one is working with a low number of observations and feature extraction needs to be automated.
- Host variables have the highest predictive power, however, due to the need of performing public database lookups, their calculation time can be a factor of 10^5 higher compared to lexical features. Combined with the fact that yield rate of usable host variables is only 30% and decreases over time, this class of predictors is difficult to recommend unless one is extracting features nearly in real-time.
- Ensemble learning methods, in particular, Random Forest and Bagging Classifier have achieved the lowest false negative rates for all 3 classes of predictors, with the peak value being a mere 1.7%. They also seem to scale exceptionally well on the datasets as large as 6×10^5 observations. However, for anything larger, Deep Learning is recommended due to its power to extract more insights and benefit from hardware acceleration.

References

[1] S. Sinha, M. Bailey, and F. Jahanian, “Shades of grey: On the effectiveness of reputation-based ‘blacklists,’” *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, Oct. 2008, doi: 10.1109/malware.2008.4690858.

[2] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, “Detecting Malicious URLs Using Lexical Analysis,” *Network and System Security*, pp. 467–482, 2016, doi: 10.1007/978-3-319-46298-1_30.

[3] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Learning to detect malicious URLs,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–24, Apr. 2011, doi: 10.1145/1961189.1961202.

[4] “The components of a URL,” *www.ibm.com*. https://www.ibm.com/docs/en/cics-ts/5.3?topic=concepts-components-url (accessed Apr. 25, 2022).

[5] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists,” *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’09*, 2009, doi: 10.1145/1557019.1557153.

[6] P. Likarish, E. Jung, and I. Jo, “Obfuscated malicious javascript detection using classification techniques,” *IEEE Xplore*, Oct. 01, 2009. https://ieeexplore.ieee.org/document/5403020 (accessed Apr. 25, 2022).

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: The Mit Press, 2016.

[8] “SecurityTrails | Whois Lookup: Definition and Examples,” *securitytrails.com*. https://securitytrails.com/blog/whois-lookup (accessed Apr. 13, 2022).

[9] “What is Whois Information and Why is it Valuable? | DomainTools Support,” *DomainTools*. https://www.domaintools.com/support/what-is-whois-information-and-why-is-it-valuable#:~:text=Whois%20is%20a%20widely%20used (accessed Apr. 13, 2022).

[10] “Thesis – Google Drive,” *https://drive.google.com/drive/folders/1vPN6aMA_Vjc0myHjeFsweuZYV9tCHizQ?usp=sharing*

[11] C. Chio and D. Freeman, *Machine learning and security : protecting systems with data and algorithms*. Beijing: O'reilly, 2018.

[12] S. Bird, *Natural language processing with python.* O'reilly Media, 2016.

[13] D. Sahoo, C. Liu, and S. C, "Malicious URL Detection using Machine Learning: A Survey," *arXiv.org*, 2017. https://arxiv.org/abs/1701.07179

[14] F. Vanhoenshoven, G. Napoles, R. Falcon, K. Vanhoof, and M. Koppen, "Detecting malicious URLs using machine learning techniques," *2016 IEEE Symposium Series on Computational Intelligence (SSCI),* Dec. 2016, doi: 10.1109/ssci.2016.7850079.

[15] Google, "Colaboratory – Google," *research.google.com*. https://research.google.com/colaboratory/faq.html

[16] A. C. Müller and S. Guido, *Introduction to machine learning with Python : a guide for data scientists*. Beijing: O'reilly, 2017.

[17] Aurélien Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc., 2019.

[18] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning : with applications in R*. Springer, 2013.

[19] "1.5. Stochastic Gradient Descent," *scikit-learn*. https://scikit-learn.org/stable/modules/sgd.html#:~:text=Stochastic%20Gradient%20Descent%20(SGD)%20is

[20] "urllib.parse — Parse URLs into components — Python 3.10.4 documentation," *docs.python.org*. https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse (accessed Apr. 25, 2022).

[21] "sklearn.preprocessing.StandardScaler — scikit-learn 0.21.2 documentation," *Scikit-learn.org,* 2019. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[22] "sklearn.metrics.fbeta_score," *scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html#:~:text=The%20F%2Dbeta%20score%20is (accessed Apr. 25, 2022).

[23] R. E. Ikwu, "Extracting Feature Vectors From URL Strings For Malicious URL Detection," *Medium,* Aug. 20, 2021. https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

[24] "(PDF) Deep learning in business analytics and operations research: Models, applications and managerial implications," *ResearchGate*.

https://www.researchgate.net/publication/336078673_Deep_learning_in_business_analytics_and_operations_research_Models_applications_and_managerial_implications (accessed May 15, 2022).

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Papuna Abesadze

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Statistical Learning Methods for Malicious URL Detection", supervised by Toomas Lepikult
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

24.03.2022

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.