Machine Learning for Malicious URL Classification: A Temporal Analysis

by Evan N. Wehr

B.S. in Computer Science, May 2017, Geneva College
B.S. in Applied Mathematics, May 2017, Geneva College
M.S. in Information Security Policy and Management, May 2019,
Heinz College of Carnegie Mellon University

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

January 5, 2024

Praxis directed by

Linda Rodriguez
Professorial Lecturer of Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that Evan Nicholas Wehr has passed the Final Examination for the degree of Doctor of Engineering as of November 30, 2023. This is the final and approved form of the Praxis.

Machine Learning for Malicious URL Classification: A Temporal Analysis

Evan N. Wehr

Praxis Research Committee:

Linda Rodriguez, Praxis Director, Professorial Lecturer of Engineering and Applied Science

Walid Hassan, Committee Member, Professorial Lecturer of Engineering and Applied Science

Raymond Chow, Committee Member, Professorial Lecturer of Engineering and Applied Science

Dedication

The author wishes to dedicate this work to his late grandfather, Ralph Wehr.

Acknowledgements

Several thanks are given:

1. My employer, The Boeing Company, who has helped to finance my Doctoral education at George Washington University.
2. A friend and coworker from The Boeing company, Ka Ip, for helping me to wrestle with APIs.
3. A friend and coworker from The Boeing company, Chace Wilcoxson, for bouncing back and forth some data science and ML ideas. In particular, ideas related to feature selection.
4. A friend and academic peer from Carnegie Mellon University, Ravneet Singh, for bouncing back and forth some data science and ML ideas. In particular, ideas related to feature selection.
5. Kara Lee Mantinaos, for serving as the technical editor of this work.
6. My parents, Larry and Becky Wehr, who encouraged their Curious George to march to the beat of his own drum.

Abstract of Praxis

Machine Learning for Malicious URL Classification: A Temporal Analysis

When Machine Learning (ML) is applied to labelled datasets, it is standard practice that 70% of the dataset is used for training models, with 30% used for testing. That approach assumes the dataset does not change over time. This praxis makes the case that URLs do change over time, meaning that ML performance should decay over time when applied to malicious URL classification. All studies found within the scope of this praxis, where ML was used to classify malicious URLs, used the traditional 70-30 split. Not knowing the consequences of drift and decay where ML is applied to malicious URL classification was identified as the primary knowledge gap of interest for this praxis.

Models that are resistant to performance decay over time exhibit lasting power. To identify models with the strongest lasting power, a dataset was created to address the stated research questions and hypotheses, which relate to the idea that a temporal analysis can determine which ML algorithms struggle the least with drift and decay when used to classify malicious URLs. The new, temporally segmented dataset had 2,292,882 URLs, making the dataset of this praxis one of the largest malicious URL datasets to date.

The temporal analysis did, indeed, reveal concept drift. Additionally, the analysis revealed that performance decay might exist, but the decay would be negligible. The presence of performance decay remains an open question that future research could explore. Models with the strongest lasting power identified herein included SVM-l, SVM-rbf, and LSVC, with the application of Normalization and Standardization. These findings provide essential novel insight so that security engineers can make informed

decisions by selecting models with lasting power when designing security solutions for malicious URL classification.

Declarations

1. The dataset of this praxis is available on request for academic purposes. You may contact the author for access at enwehr@gmail.com, or LinkedIn at www.linkedin.com/in/evanwehr/.
2. Any graphics, figures, or tables herein were reproduced with permission from at least one author of the original source works.
3. The author declares no conflict of interest.

Table of Contents

List of Figures

List of Tables

List of Symbols

$\frac{X}{Y}$	Division sign (Dividing X by Y)
X/Y	Division sign (Dividing X by Y)
[ ]	Brackets, denoting a matrix
=	Equals sign
i	Index
Ranked( )	Function to transform a variable into its ranked order position within the variable set.
Max( )	Function to select the maximum number within a variable set
−	Minus sign
x	Multiplication sign
n	Number
N	Number of samples within a category of Welsh's ANOVA
$\hat{y}$	Predicted value of y
α	Significance level
$H_{i\text{-}n}$	Subscript i denotes the index of the hypothesis set (H), and subscript '-n' denotes this as the set's null hypothesis.
$H_{i\text{-}a}$	Subscript i denotes the index of the hypothesis set (H), and subscript '-a' denotes this as the set's alternate hypothesis.
R_s	Spearman's Correlation Coefficient
∑	Summation (sigma)

List of Acronyms

AI	Artificial Intelligence
AML	Adversarial Machine Learning
ANOVA	Analysis of Variance
APT	Advanced Persistent Threat
AUC	Area Under the Curve
Bi-LSTM	Bidirectional Long Short-Term Memory
BN	BayesNet
CB	Categorical Boosting
CBA	Classification Based on Association
CM	Confusion Matrix
CNN	Convolutional Neural Networks
CSNN	Cost-Sensitive Neural Networks
DNN	Deep Neural Networks
DT	Decision Tree
ET	Extra Trees
F1	f1-score
FN	False Negative
FP	False Positive
FFNN	Feed Forward Neural Networks
GNB	Gaussian Naïve Bays
GMM	Gaussian Mixture Model
H	Hypothesis

HEOM	Heterogeneous Euclidean-Overlap Metric
HTML	HyperText Markup Language
IDS	Intrusion Detection System
IBM	International Business Machine Corporation
IndRNN	Independently Recurrent Neural Networks
IETF	Internet Engineering Task Force
KMeans	KMeans Clustering
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LGBM	Light Gradient-Boosting Machine
LMT	Logistic Model Tree
LSTM	Long Short-Term Memory
LR	Logistic Regression
ML	Machine Learning
MLP	Multilayer Perceptron (H denotes hidden layer count if used)
NB	Naïve Bayes
NN	Neural Network
OSINT	Open Source Intelligence
PART	Partial C4.5
QDA	Quadratic Discriminant Analysis
RBF	Radial Basis Function
RF	Random Forests
RILB	Raced Incremental LogitBoost

RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
RQ	Research Question
RT	Random Tree
SDL	Sequential Deep Learning
SGD	Stochastic Gradient Descent
SMO	Sequential Minimal Optimization
SMOTE	Synthetic Minority Oversampling Technique
SVC	Support Vector Classification
SVM	Support Vector Machines
td-idf	Term Frequency-Inverse Document Frequency
TLD	Top Level Domain
TN	True Negative
TP	True Positive
URL	Uniform Resource Locator
VP	Voted Perceptron
WRST	Wilcoxon Rank-Sum Test
XGB	eXtreme Gradient-Boosting

List of Equations

Chapter 1 - Introduction

1.1 Background

Since its conception in the early 1990s, the internet has grown rapidly in scale. Security was an afterthought. While the internet has transformed society, so too has the internet transformed crime. Phishing is a primary weapon in said criminal efforts in which users are deceived over email and lured to click on links or open attachments that are not safe, meaning they can: (a) send malware to the user, (b) steal credentials or private information, or (c) deceive the user by presenting fraudulently-altered web content.

IBM's *Cost of a Data Breach Report 2022* states that the average global cost of a data breach to be $4.35 million, with the average cost in the United States alone being $9.44 million (IBM, 2022, p. 5, 7). These numbers are at an all-time high, with phishing being the cause of 16% of these breaches (IBM, 2022, p. 5, 6). IBM's calculations account for the costs of notifications to users and regulatory compliance groups, post-breach response, detection and escalation, and lost business. Though there could be consequences to phishing other than just data breaches, and though there are malicious links beyond those used in phishing emails, security controls exist to counter the risk of malicious links.

There are three traditional preventative security controls for blocking access to malicious URLs: blacklisting-based, anomalies-based, and heuristic-based (Aljabri et al., 2022). However, each approach has shortcomings. Blacklisting approaches will only catch malicious URLs that are known and will only block exact matches. Anomaly

approaches can be challenging to implement effectively, as anomalous features do not guarantee maliciousness and may lead to high rates of false positive blocks. Heuristic approaches will build a portfolio of known attack patterns, but as with blacklisting, new attacks can avoid the heuristic patterns.

To compensate for these shortcomings, cybersecurity research in recent years has focused on the use of Machine Learning (ML) to develop models that can classify URLs as malicious or benign. ML is a subset of the field of study known as Artificial Intelligence (AI) that uses algorithms to imitate human learning. Applied to malicious URLs, the objective of ML is to produce a statement of confidence about the safety of a URL, and then classify the URL based on a confidence threshold. To this end, this praxis studies the use of ML to classify malicious URLs. Lexical features will be another focus of this praxis, meaning that the malicious URL dataset will be based on the statistical properties of the URLs. Furthermore, and of primary importance, is that this praxis will use a temporal analysis, which deviates from all known prior research of ML applied to malicious URL classification. A temporal analysis for ML applied to malicious URL classification is novel to this praxis.

1.2 Research Motivation

When ML is applied to labelled datasets, it is standard practice that 70% of the dataset is used for training a model and 30% is used for testing the model. That approach is fine as long as the dataset does not evolve over time. Most ML algorithms are designed with the assumption that data is from stationary populations and does not change over time; however, “nonstationary populations are said to exhibit concept drift” (Singh, 2012, p. 6). Concept drift means that the features of data change over time, and if concept drift

is present, the consequence would be that a model's performance should decay (decrease) over time. Therefore, this praxis assumes that malicious URLs change over time.

In this regard, the key inspiration to this praxis is *BODMAS: An Open Dataset for Learning based on Temporal Analysis of PE Malware* (Yang et al., 2021). Instead of taking the traditional 70-30 split approach, Yang's research attention was given to how malware evolves over time. Yang selected a point in time where everything prior to that point in time was a candidate for training models, and everything after that point was a candidate for testing models. Using month-long testing intervals, performance plots showed accuracy decay for some models, while other models showed resistance to accuracy decay. One of Yang's performance charts has been included in Figure 1. Temporal plots such as this display performance decay over time, while 70-30 split approaches cannot do so. Thus, this praxis will refer to models that exhibit resistance to performance decay as models that possess lasting power.

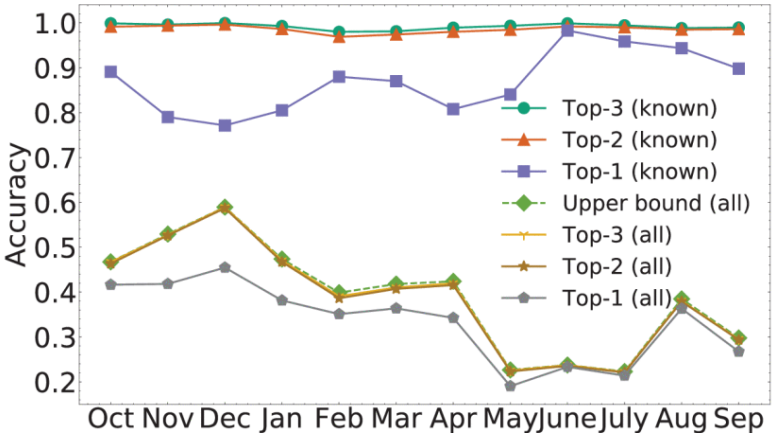


Figure 1: A month-wise temporal performance plot (Yang et al., 2021)

1.3 Problem Statement

All studies examined within the scope of this praxis used the traditional 70-30 split when using ML to classify malicious URLs. Although some research has admitted

concern for concept drift in malicious URL classification, none accounted for concept drift and none performed a temporal analysis. Not knowing the possible consequences of how ML models perform over time when applied to malicious URL classification could lead to suboptimal models being selected for use which do not possess lasting power.

1.4 Thesis Statement

Further insight is needed on how to determine which ML models should be selected for malicious URL classification beyond the insight gained from traditional 70-30 splits. Model selection needs to account for the possibility that URLs change over time. If URLs are nonstationary, models used for classifying malicious URLs should exhibit concept drift and, consequently, should decay in performance over time. Neglecting to evaluate the implications of performance decay for ML applied to malicious URL classification is a mistake and constitutes malpractice to security practitioners. A temporal analysis can determine which ML algorithms struggle the least (or most) with drift and decay when applied to malicious URL classification.

1.5 Research Objectives

Following the thesis statement's aim, the work of this praxis was oriented to the following Research Objectives (RO):

RO_1: Produce a dataset adequate for a lexical temporal analysis of URLs labelled as either malicious or benign.

RO_2: Produce insight that can serve to guide future research on ML applied to malicious URL classification.

RO_3: Produce insight that can be used by security engineers and data scientists, such that they can make informed decisions on selecting models with lasting

power when they are designing security solutions for malicious URL classification.

1.6 Research Questions and Hypotheses

The following Research Questions (RQ) guided the exploratory work of this praxis, and the following Hypotheses (H) were candidates for evaluation in this praxis:

1.6.1 Research Questions

RQ_1: Over time, do ML models trained to classify malicious URLs exhibit decay or drift in performance?

RQ_2: For malicious URL ML models that demonstrate drift or decay, are there models with stronger lasting power?

RQ_3: Given enough time for a model's performance to drift or decay, do we observe the rank-ordering of performance change between models?

RQ_4: Can rescaling data improve or reduce drift or decay in performance?

1.6.2 Hypotheses

Hypotheses map to the prior listed research questions. Null Hypotheses are denoted with subscript '-n', with the alternate hypotheses denoted with subscript '-a'.

$H_{1\text{-}n}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that all models exhibit no drift and no decay in performance over time.

$H_{1\text{-}a}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that there are models with drift or decay in performance over time.

$H_{2\text{-}n}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that all models exhibit equal lasting power.

$H_{2\text{-}a}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that there are models that exhibit different lasting power.

$H_{3\text{-}n}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that all models do not change in their performance rank-ordering.

$H_{3\text{-}a}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that models do change in their performance rank-ordering.

$H_{4\text{-}n}$: Of ML models created to lexically classify malicious URLs, there is no temporal performance difference among different data rescaling techniques.

$H_{4\text{-}a}$: Of ML models created to lexically classify malicious URLs, there is temporal performance difference among different data rescaling techniques.

1.7 Scope of Research and Assumptions

As mentioned, this praxis delivers a lexical temporal analysis. Lexical analysis is introduced in section 2.5. The dataset of this praxis, as discussed in section 3.1, relies on publicly available datasets and Open Source Intelligence (OSINT). Section 2.5.6 provides further information on why it is desirable to include data from publicly available OSINT, but the assumption is that OSINT data is more likely to exhibit drift and decay. Further, URLs (malicious or not) are non-stationary and consequently exhibit drift, reinforced in sections 2.4, 2.5.1 and 2.5.2.

1.8 Research Limitations

Future research seeking to replicate findings herein should consider the following:

1.8.1 Time

This praxis was worked entirely in 2023. Due to that pacing, the temporal analysis was less than a year in duration. Specifically, 16 test buckets of one-week intervals were used. It is possible that a longer temporal analysis could show different results.

1.8.2 Possibility of Data Biases

Reliance on OSINT and publicly available data brings an increased likelihood of unknown bias within the dataset. Speaking to malicious URLs for the 16 test buckets, URLHaus provided the sole source. Though other sources were considered, they were forgone due to either not being available, or being unaffordable.

Regarding benign URLs for the 16 test buckets, URLs with the date that they were reported could not be obtained within the time constraint of this praxis. While the malicious URLs of the temporal analysis are stratified by the date they were reported to URLHaus, a random sample of benign URLs were removed from the training set and moved into test buckets. Section 3.2.2 will review this in greater detail and why this was deemed acceptable.

1.9 Organization of Praxis

Chapters herein are organized to transfer knowledge to readers as follows:

Chapter 1 - Introduction: This chapter introduces readers to the scope of this praxis.

Chapter 2 - Literature Review: This chapter reviews background information and related works of research.

Chapter 3 - Methodology: This chapter reviews how the lexical dataset was constructed and provides detail on how machine learning was accomplished on said dataset.

Chapter 4 - Results: This chapter shares performance measurements of the temporal analysis and evaluates hypotheses and research questions.

Chapter 5 - Discussions and Conclusions: This chapter reviews implications of findings herein, provides emphasis for contributions to the body of knowledge, and lists recommendations for future works of research.

Chapter 2 - Literature Review

2.1 Introduction

In recent years, ML has gained the attention of researchers for numerous applications across many different industries, including cybersecurity. This Literature Review will: (a) review the subject of AI and common algorithms used for classifying malicious URLs in prior research; (b) review challenges of ML for URL classification; and, (c) review a selection of previous research.

2.2 Review of Artificial Intelligence

For decades, there have been efforts "to make machines do intelligent things" without human direction, and this field of study is known as Artificial Intelligence, or AI for short (Xiao, 2022, p. 3). In AI, computers consume an input of information and learn to take autonomous actions. The aim of AI is to solve problems that are potentially challenging for people to solve manually at scale but are more easily solved autonomously by a computer at scale. AI can be hierarchically organized by subsets of study. As shown in Figure 2, Deep Learning (DL) is subset to Neural Networks (NN), NNs are subset to ML, and ML is subset to AI (IBM Data and AI Team, 2023).

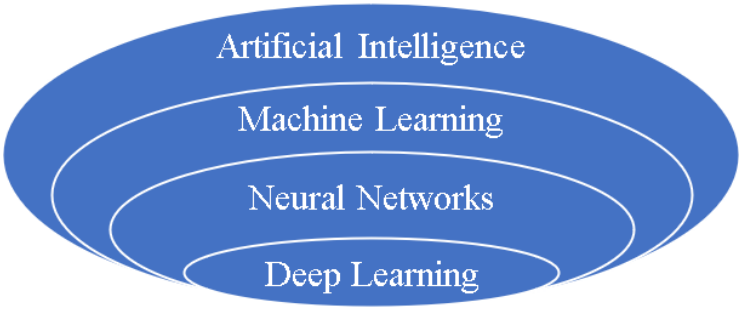


Figure 2: Venn Diagram of Artificial Intelligence Domains (IBM Data and AI Team, 2023)

From the 1950s to 1970s, artificial Neural Networks were developed to mimic how biological neural networks learn (i.e., the human mind) (Xiao, 2022, p. 6). Neural Networks typically have three layers: an input layer, a hidden layer, and an output layer. The computer in this case is left with determining how to map that hidden layer. Following this, due to less than promising results, Machine Learning rose in popularity in the 1980s to 2010s as a less computationally cumbersome alternative to Neural Networks. Machine Learning relies on mathematical algorithms to analyze data. From the 2010s onward, Deep Learning has been the focus of Neural Network research. Deep Learning methods are Neural Networks with multiple hidden layers, made possible with increased computing power and better graphical processing units (Xiao, 2022, p. 7). The trend toward cloud computing has also made the study of Deep Learning and Neural Networks more feasible, as access to higher-end computing resources could be utilized ad-hoc and paid for by usage.

To solve a problem, ML requires a dataset, or collection of information, from which to learn. From the dataset, ML will produce what is known as a model, where the model serves to handle autonomous solution-finding. In this effort, traditionally speaking, it is standard practice to randomly divide a dataset in two, where one portion of the dataset becomes the training dataset and the other becomes the testing dataset (Brownlee, 2020). The training dataset is used by ML for fitting (i.e., training) a model and the testing dataset is used for evaluating how well the model performs when solving a problem. The ratio between training and testing is commonly done at a 70% training and 30% testing split, or at least with the training dataset being larger than the testing dataset (Shah, 2021).

2.3 Machine Learning Methods

Models produced by ML will vary depending on what ML algorithm was used to produce the model. Algorithms are categorized as one of three functional learning methods: supervised learning, unsupervised learning, and reinforcement learning (Delua, 2021).

Supervised learning is synonymous with “labeled” learning, where the dataset comes with a label for each element and the model’s objective is to predict what the label should be for test elements (Glassner, 2021, p. 9). While the label is included within the training dataset, the model does not know what the label is within the testing dataset; performance is measured by how well the model can predict what the testing dataset label should be. Image classifiers, weather prediction, and stock market predictions are good examples of problems suitable for supervised learning. Supervised learning is used for classification and regression problems and is called supervised because it is reliant on an external supervisor, such as a data scientist, to provide the label. Not all datasets come with labels, though, and providing labels might not be feasible, depending on the problem.

In contrast, unsupervised learning has no label and takes its name from the fact that no supervisor is needed for the algorithms to work. Rather than predicting the label for input elements, unsupervised learning tries to predict relationships between input elements (Glassner, 2021, p. 8). As such, unsupervised learning is used for clustering and association problems. Evaluating performance of an unsupervised model is less straight-forward than supervised learning and relies to a greater extent on incrementally comparing and contrasting alternative models statistically. For example, imagine an

archaeological dig site where thousands of broken ceramic jars have been found and no one can determine how to organize the broken pieces. An unsupervised algorithm might be able to sensibly cluster or group the pottery shards.

Reinforcement learning, furthermore, is good for solving reward-based problems using trial-and-error methods. In reinforcement learning, incentives and penalties for the algorithm are defined, with the goal of having the model reach a desirable end state. An example of reinforcement learning would be to play Super Mario Bros, finding the end of the level (the end state), while maximizing coins collected and minimizing time (Heinz, 2019). After enough attempts, the reinforcement algorithm will produce models better at reaching those criteria.

2.4 URL Anatomy and Datasets

A Request For Comment (RFC) is a documented specification for how a technology works and how a technology should be implemented; it is used by the Internet Engineering Task Force (IETF) and other standardizing organizations. Uniform Resource Locators (URLs) are defined in RFC 1738 and RFC 3986 (Berners-Lee et al., 1994; Berners-Lee et al., 2005), and were published in 1994 and 2005, respectively. URLs are a fundamental building block of the internet, defining "a compact string representation for a resource available on the Internet" (Berners-Lee et al., 1994). In other words, a URL is an address where a website can be found. This makes URLs a universally-shared feature among all websites. Websites, however, as the destinations of URLs, are collections of information, and are usually designed for visual display using HyperText Markup Language (HTML) and a combination of other programming languages. For this reason,

‘URL’ and ‘website’ are often used interchangeably; ‘malicious URL classification’ and ‘malicious website classification’ often refer to the same thing, as well.

Datasets designed for classifying malicious URLs can be composed in several different ways, focusing either on one, or a combination of: (a) the HTML content (i.e., the code) of the URL destination; (b) the network traffic characteristics going to and from the URLs, which tend to be oriented for intrusion detection (or prevention) systems; and, (c) statistical characteristics of the URL’s textual features, which would be a lexical based dataset. All of these could additionally pull in information from third-party external sites such as VirusTotal or Whois, but a primary shortcoming with relying on external sources is that websites change over time. In fact, malicious websites often only exist for a few minutes at a time and can change their registration information and HTML content. Additionally, “many malicious URLs are not always alive, and extracting their features may not be possible after their expiration” (Swarnkar et al., 2022, p. 214). This can complicate how consistently (and thereby reliably) external sources can be used for validating malicious URLs, given enough time for their features to change.

This evolutionary nature of websites can make the HTML-based approach listed above challenging to validate or to replicate results of prior research. Datasets for HTML-based learning can be converted to either: (a) a lexical analysis of the website’s code; or, (b) a visual analysis of how the websites look. Datasets for HTML-based learning often include a URL, in which case the dataset can be repurposed for lexical based analysis of the URL, instead of the HTML. This is another factor as to why lexical URL analysis is a popular option, compared to a lexical HTML analysis in malicious URL classification

research. It is for that reason that this praxis is also oriented toward a lexical URL analysis.

Looking further into URL anatomy, a simplified sample URL is broken down in Figure 3. This figure has been organized based on syntax definitions between RFCs 1738 and 3986 (Berners-Lee et al., 1994; Berners-Lee et al., 2005), and samples provided by Crişan et al. (2020, p. 189), Bahnsen et al. (2017, p. 2), and developer documentation from Mozilla (*What Is a URL? - Learn Web Development*, n.d.). The list of explanatory terms following the figure is also taken from these sources.

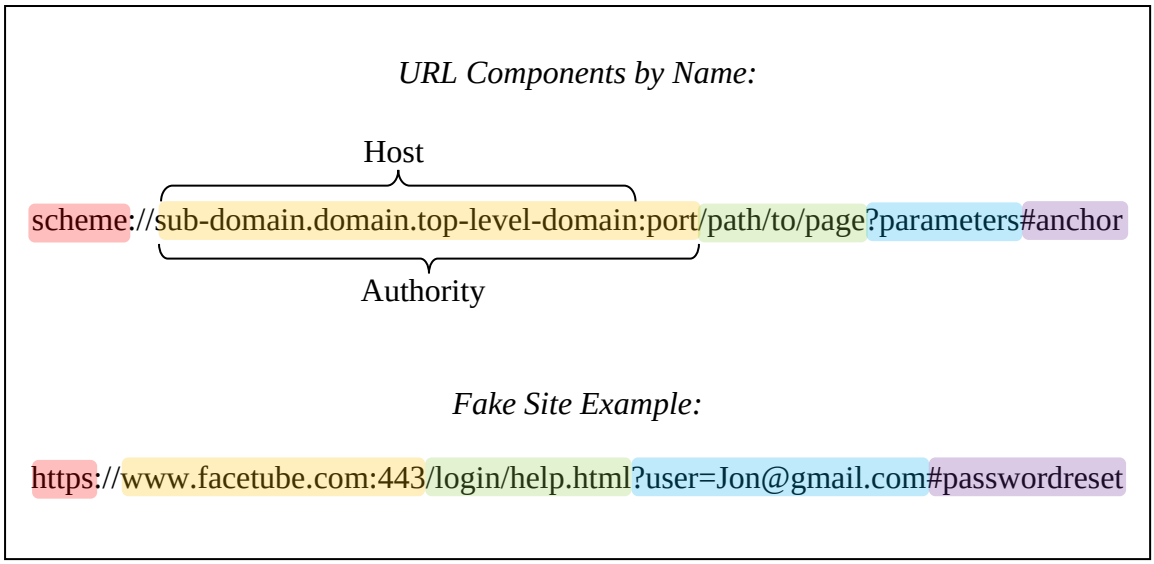


Figure 3: URL Syntax

- **Scheme**: Also known as protocol or prefix, the scheme is used to request the URL. Though scheme can be left undeclared, using ‘http’ uses no encryption in transmission, and ‘https’ uses encryption. If undeclared, ‘://’ will also be omitted, which is only used to separate the scheme from the rest of the URL. Many secure websites will convert URLs to https, if possible, as a security best practice, but

alternate schemes include ftp, gopher, mailto, news, nntp, telnet, wais, file, and prospero.

- **Authority**: Separated from the scheme by '://', the authority contains a domain name and a port number separated by a colon.
- **Host**: The host defines the web server being requested by the URL. Alternatively, an IP address can be used. The host is converted to an IP address upon use, with the host serving as a human-friendly feature of URLs. This is to say that the host is used because they are more easily read and rememberable than IP addresses. Host components include the sub domain, domain, and top-level domain.
- **Port**: The port defines which gate is used on the web server. If undeclared, most web traffic defaults to port 80 for HTTP, and 443 for HTTPS.
- **Sub Domain**: Sub domain is located to the left of the domain and separated by a full stop '.', also called 'dot'. "www" is used by default, which stands for World-Wide Web, and can be left undeclared, but there are many valid alternates.
- **Domain**: The domain is the name of the website being requested.
- **Top Level Domain**: Top Level Domain, or TLD for short, is to the right of the domain and separated by a full stop '.'. Most often, '.com' is used.
- **Path**: The path functions to declare the physical file location of the web page of the website to be returned, which is stored on a web server.
- **Parameters**: Parameters are an optional means to pass information to the web server. Individual parameters are prepended with a '?' and are separated from each other with '&'s.

- **Anchor**: Anchors are bookmarks of sorts, and if used by the web page, will guide a web browser to the section of the web page where the anchor is defined.

2.5 Machine Learning Challenges in Lexical URL Analysis

As mentioned in section 2.4, registration information of websites can change rather quickly, making information returned by external third party sources such as VirusTotal and Whois inconsistent across time; however, that is only a single troubling obstacle to lexical machine learning research. There are numerous challenges to review.

2.5.1 Concept Drift

Most machine learning algorithms are designed with the assumption that training data is from stationary populations not changing with time; however, "nonstationary populations are said to exhibit concept drift" (Singh, 2012, p. 6). Considering that the internet (and its websites) do indeed evolve with time, it can be said that the characteristics of malicious and non-malicious URLs are non-stationary and present a risk to URL classification models. Here, the concern is that "the rules and patterns recognized by the model may become obsolete as the environment changes" (Castillo, 2023).

Joao Gama outlines two types of drift (real and virtual) in his work *A Survey on Concept Drift Adaptation*. Reproduced in Figure 4 is a visual explanation of these two types described by Gama (2014, p. 5). Real concept drift happens when the characteristics of a class shift, affecting boundary of difference between class. Virtual drift occurs when characteristics of elements change without affecting the boundary of difference between class. When considering time, there are several ways to describe drift, which Gama also describes, reproduced in Figure 5, (2014, p. 5). Challenges here

concern how often models should be retrained, selecting models that perform well over time, and ensuring that drift-detection techniques do not conflate outliers and drift, while also catching low-frequency attacks.

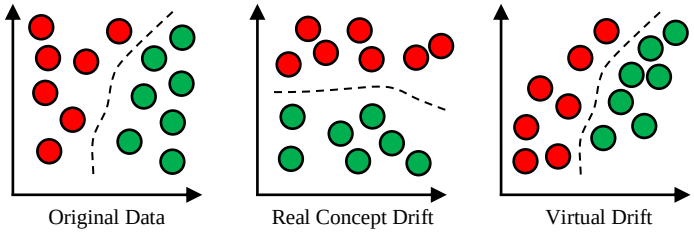


Figure 4: Concept Drift Visualized (Gama et al., 2014, p. 5).

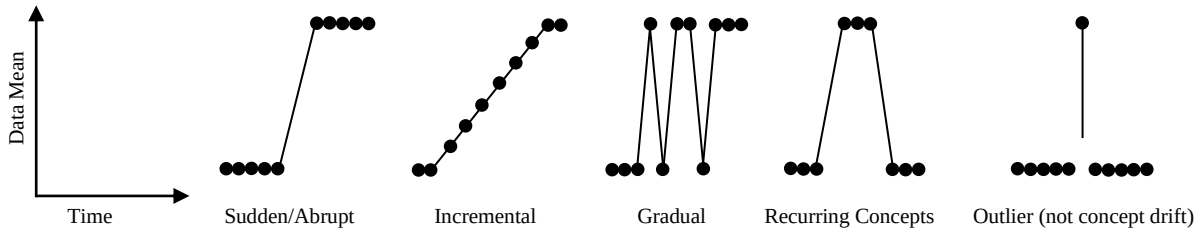


Figure 5: Temporal Drift Descriptions (Gama et al., 2014, p. 5).

2.5.2 Volume, Velocity, Variety

The internet has grown exponentially since conception. At the time of writing, worldwidewebsize.com estimates there are at least 6.76 billion web pages (*The Size of the World Wide Web (the Internet)*, 2023). Physical internet infrastructure links commonly operate at speeds of 100Gbps or higher, and to the point of volume, these links can handle more than 148,809,524 packets per second (*CEC Juniper Community*, 2009). Further, many websites are dually designed to account for the diversity of different device types that traffic to web pages, which can add additional overhead to the variety of normal and malicious websites (Shone, 2018, p. 2). With new URLs created daily to keep up with the growth of the internet, URLs cannot be stationary.

Considering the scale at which the internet operates, and the numerous functions the internet serves, the task of training a model for handling malicious URLs for the entire internet is nearly impossible (Swarnkar et al., 2022). Compensating for this challenge, research on URLs has taken the approach of sampling a subset of the internet, but a further question remains as to whether there is bias or not within the sampled subset. As a rule of thumb, larger datasets from additional sources are generally preferrable.

2.5.3 Acquiring Labels, and Label Accuracy

In the search for data, most ML algorithms require a label to ascertain whether a URL is malicious. One option is to ask experts to manually label an unlabeled dataset. Manually adding labels to URLs can be labor intensive, adding an inverse incentive to generally wanting more URLs, and will raise a further question regarding human fallibility. Labels may also be sourced from blacklists or whitelists, pre-existing datasets, or from third party Open Source Intelligence (OSINT) sites. In all cases, a further question remains as to whether or not the labels are correct.

A challenge presents itself in validating that labels are acceptable, which most often can be addressed by validating a sample of URLs in bulk over API, against tools like VirusTotal, GoogleSafeBrowsing, and others. It is important to keep in mind, however, that APIs commonly are not free. Funding to pay for APIs adds a monetary barrier to entry depending on the volume of URLs needing to be checked.

2.5.4 URL Redirects and Obfuscation

Google defines ‘Redirecting URLs’ as “the practice of resolving an existing URL to a different one, effectively telling your visitors and Google Search that a page has a new

location" (*Redirects and Google Search*, n.d.). Redirection of URLs is an important function in order for websites to work, accomplishing: (a) temporary redirects for downtime or site maintenance; and, (b) permanent redirects for handling web page removals, preserving website accessibility (*Redirections in HTTP*, n.d.). To the point of (malicious) URL datasets, a valid question is to ask if redirects complicate or inhibit the accuracy of URL classification, as the final URL destination of both benign and malicious URLs may be different from the URL presented in the dataset.

Similarly, malicious URLs are often designed with obfuscation, "with the aim of hiding the real host, and particularly the registered domain, the only part of the URL that cannot be freely defined" (Marchal et al., 2014). Simply put, obfuscating a URL is to intentionally hide the final URL destination. On the other hand, obfuscation techniques are not always an indication of malicious URLs, as not all obfuscated URLs are malicious. The difference between obfuscation and redirects is that obfuscation is used with the intention of masking the final URL destination, while redirects are not.

Though tools like wheregoes.com can be used to confirm presence of redirects and possible obfuscation, wheregoes.com will not be able to capture the redirection that existed in the past, only seeing redirects existent at the time of check. Redirection and obfuscation remain an open concern in URL classification research; however, there is research demonstrating that lexical based analysis can overcome these obfuscation techniques (Garera et al., 2007; Marchal et al, 2014; Sabir et al., 2022).

2.5.5 Imbalance, Oversampling, Undersampling, and SMOTE

Imbalanced learning can be defined "as the learning process for data representation and information extraction with severe data distribution skews to develop

effective decision boundaries to support the decision-making process" (He & Ma, 2013, p. 1). Though there is no consensus as to what can quantifiably be considered imbalanced, "most practitioners would certainly agree that a dataset where the most common class is less than twice as common as the rarest class would be [only] marginally unbalanced," and not cause for concern (He & Ma, 2013, p. 15). Typically, in malicious URL datasets, benign URLs comprise the common class and malicious URLs make up the rare class.

The concern with imbalance is in regard to how well regression, classification, and clustering ML algorithms can learn the rare class. At a high enough degree of imbalance, algorithms may disregard the rare / malicious class, "leading some [algorithms] to ignore the minority class entirely" (Brownlee, 2021a). Some research disregards and does not address imbalance. Of research that does address imbalance, options include undersampling the common class, oversampling the rare class, applying synthetic oversampling to the rare class, or a combination of the three. In regard to which is preferable, that may further depend on which algorithm is being used, or how the dataset is composed. Undersampling and oversampling are rather intuitive. In undersampling, a random proportion of the common class is disregarded until the ratio to the rare class is commensurate; however, that comes with the risk of disregarding important features in the common class, negatively impacting model performance. In oversampling, the rare class is randomly counted in duplicate until the ratio to the common class is commensurate; however, that comes with the risk of overfitting subsequent models, where the model is overly sensitive to the features present in the training portion of the rare class, negatively impacting model performance. Synthetic

oversampling, on the other hand, can compensate partially for that risk (Brownlee, 2021a).

SMOTE, or Synthetic Minority Oversampling Technique, is "perhaps the most widely used approach to synthesizing new examples" for the minority / rare class (Brownlee, 2021b). SMOTE is not new, as it predates Scikit-Learn's 2007 release date (*About Us — Scikit-Learn 0.22.2 Documentation*, n.d.) by five years, described by Nitesh Chawla in his 2002 paper *SMOTE: Synthetic Minority Over-sampling Technique* (Chawla et al., 2002). Worded simply, "SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line," similar to KNN (Brownlee, 2021b). One concern for SMOTE is that "synthetic examples are created without considering the majority class, possibly resulting in ambiguous examples if there is a strong overlap for the classes," but that concern should be less pressing for only marginally imbalanced datasets (Brownlee, 2021b). Conversely, as cautioned by Microsoft, "SMOTE is not guaranteed to produce more accurate models" (Likebupt, 2021).

Noteworthy for each technique for handling imbalance (undersampling, oversampling, and SMOTE) in labeled datasets, is that the technique used should only affect the test portion of the dataset, and not the training portion. This should be the case because in the real world, the model will not know the actual label for what is otherwise test data. Worded differently, because real world data will not have a label, the real data cannot be treated for imbalance.

2.5.6 Adversarial Machine Learning

In *Adversarial Machine Learning for Cyber Security* De Lucia and Cotton admit that "the security of machine learning, also referred to as Adversarial Machine Learning … is not well understood within a cyber security context" (2019, p. 1). Adversarial Machine Learning, or AML for short, seeks to attack fundamental assumptions of Machine Learning solutions, such as the dataset adequately describing the underlying aspects of the problem (Muhati & Rawat, 2021, p. 1).

In the context of malicious URL classification solutions, AML can refer to either 'dataset poisoning' or 'model evasion'. In dataset poisoning, the goal of the adversary is to adversely affect the dataset used in the ML security model to increase the chance that their malicious URLs can bypass models' ability to classify URLs accurately. In effect, dataset poisoning seeks to adversely affect the integrity of the security model's performance (Yeboah-Ofori et al., 2021, p. 76). Model evasions, also known as exploratory attacks, occur when attempts are made to leverage knowledge and circumvent the security solution by targeting blind spots in the model (Huang et al., 2011, p. 52). For example, an evasion AML attack would occur when an adversary knows the dataset and algorithm used to develop the URL classification model and seeks to deceive that security solution by (a) using the same dataset and algorithm used to build a duplicate ML security model; (b) building their own ML model off of that dataset; (c) testing new candidate malicious URLs; and then, (d) selecting malicious URLs that can evade or "deceive previously trained classifiers" (Martins et al.2020, p. 35405). The effort and sophistication required to successfully implement model evasion tends to exclude all but Advanced Persistent Threats (APTs).

Methods for detecting that AML has occurred can be difficult to implement successfully, as the consequences of AML are nearly identical changes seen by concept drift in nonstationary datasets. Poisoning-based AML is one of many factors that contribute to concept drift. When it comes to third-party vendors that aggregate malicious URLs using 'the wisdom of crowds,' such as PhishTank, OpenPhish, and URLHaus, poisoning is possible by bad actors inserting fraudulent data. As such, project owners of security solutions that use data sourced from OSINT should be aware that drift may be more prevalent, in contrast to datasets that are privately sourced (Moore & Clayton, 2008). Further, attackers can also use both these open source datasets and publicly available datasets from sites such as Kaggle to assist in building evasion attacks. However, that assumes an attacker has knowledge as to which open source or public datasets are being used in the security solutions (Usama et al., 2020).

2.6 Research Related to URL Classification

When it comes to classifying malicious websites, there have been creative approaches recently, such as leveraging blockchains (Edirimannage et al., 2022), and visually contrasting abstracted graphical constructs developed from scraped HTML files (Barlow et al., 2020). Others take simpler approaches, such as statistically analyzing similarity of sites for correlations to reputation (Cui et al., 2017; Tanaka et al., 2021). Some research looks to analyze malicious URLs specific to platforms, such as phishing on Twitter (Bell, 2019; Roy, 2021), or analyzing the volume of phishing URLs based on impersonated industries so as to anticipate which industries may be impersonated in the near future (Alarifi, 2013; Dobolyi & Abbasi, 2016;). As this praxis uses a lexical analysis, the works of research of primary interest are those that perform lexical analysis.

2.6.1 Lexical-based Research in URL Classification

Table 1 captures a list of noteworthy works of research on the use of machine learning for URL classification where lexical features are the primary focus. Though spaced across nearly a decade, these works can be considered contemporary research for both contrast to this praxis, and general discussion. They further serve as minor sources of inspiration, second to *BODMAS* (Yang et al., 2021). Following is a general review of commonalities.

Table 1: Related Research on ML URL Classification Based on Lexical Features

Year of Publication	Author in Bibliography	Data	Models	Best Performer
2022	Abuadbba et al.	100,000 (PhishTank, OpenPhish, Alexa Topsites Amazon service)	LR, DT, KNN, NB, RF, SVM	LR
2022	Aljabri et al.	66,506 URLs subset of Malicious and Benign Websites Dataset, (A. K. Singh, 2020)	NB, RF, LSTM, CNN	NB
2022	Ghaleb et al.	651,191 URLs (Malicious Urls Dataset on Kaggle, souring from PhishTank and ISCX-URL2016)	NB, LR, DT, RF, CNN, SDL	RF
2022	Oshingbesan et al.	Over 2 million URLs from 16 pre-existing datasets.	LR, SVM, DT, RF, CB, KNN, FFNN, NB, GMM, KMeans	KNN
2022	Sabir et al.	193,386 URLs (Dmox, ISCXURL2016, Alexa web-crawl, Phish-storm, PhishTank, OpenPhish)	RF, DT, SVM, LR, KNN, XGB, LGBM, CNN, LSTM	Mixed results. Varied performance based on features.
2021	Aalla, et al.	420,000 URLs (sourcing undeclared)	DT, LR	LR
2021	Rupa et al.	32k URLs (PhishTank, Github, Kaggle)	LR, SVM, KNN, LDA	LR
2021	El-Din et al.	63,191 URLs (sourcing undeclared)	LR, DT, NB	DT
2021	Hannousse et al.	11,430 URLs (Alexa, Yandex, PhishTank, OpenPhish, & Şahingöz et al., 2019)	DT, RF, LR, NB, SVM	RF
2021	Kumi et al.	1,200 URLs (Alexa, OpenPhish, VxVault, URLhaus)	CBA, SVM, LR, NB	CBA
2021	Fan	21,615 URLs (PhishTank, others)	NB, SVM, LR, NN	NN
2021	Mitra & Raja M.V.	165,366 URLs (OpenPhish, MBPI, malware domain dataset, WEBSPAM-UK2007, ISCX-URL2016 (Mamun et al., 2016))	KNN, RF, SVM	RF
2021	Shantanu et al.	450,000 URLs (Kaggle)	LR, SGD, RF, SVM, NB, KNN, DT	RF
2021	Yuan et al.	65, 927 URLs (Alexa, host-file.net, PhishTank)	A new model, and NNs (Attention Mechanism, CapsNet, CNN, LSTM, Bi-LSTM, IndRNN, Bi-IndRNN)	A new model using an Attention Mechanism, Bi-IndRNN, CapsNet
2020	Crişan et al.	Undeclared volume of URLs (an anonymous Romanian cyber security company)	CSNN, MLP, ET	MLP
2020	Das et al.	194,798 URLs (open data sources)	Simple RNN, Simple LSTM, CNN-LSTM	CNN-LSTM
2020	Li et al.	331,622 URLs (PhishTank, 360.com, Hao123, DMOZ)	KNN, RBF Network, RBF Classifier, Kernalized-SVM, Linear-SVM, LDA, LR, MLP (H1, H3, H7),	KNN, RBF Classifier, and NB

			CNN, QDA, NB, VP, AB, RILB	
2020	Shahrivari et al.	Apx 11,000 URLs (PhishTank and other non-declared sources).	LR, DT, RF, AdaBoost, KNN, NN, SVM (linear, poly, rbf, and sigmoid), GB, XGB	XGB
2020	Srinivasan et al.	Set 1 of 611,894 URLs (Alexa, DMOZ, MalwareDomainlist, MalwareDomains, CEN Amrita Vishwa Vidyapeetham) Set 2 of 124,574 URLs (Sophos)	RNN models (Endgame, CMU Architecture), CNN models (NYU Architecture, Invincea Architecture), & hybrid MIT Architecture.	Mixed results
2020	Wei et al.	20,208 URLs (Common Crawl dataset, PhishTank, Moz Top 500, Alexa)	RF, SVM, LSTM, CNN, CANTINA+, td-idf,	CNN
2020	Xuan et al.	470,000 URLs (PhishTank, URLhas, Alexa, Malicious_n_Non-Malicious URL)	SVM, RF	RF
2019	Brites & Wei	17,381 URLs (Alexa Top Sites, PhishTank, OpenPhish)	RF, KNN, SVC, DT, MLP, GNB, Adaboost, LR	RF and MLP
2019	Şahingöz et al.	73,575 URLs (PhishTank, Yandex	DT, AdaBoost, Kstar, KNN, RF, SMO, NB	RF
2018	Jiang et al.	7 million URLs (Phish Tank, Virus Total, Google, DMOZ, and others)	CNN	CNN
2019	Şahingöz et al.	73,575 URLs (PhishTank, Yandex	DT, AdaBoost, Kstar, KNN, RF, SMO, NB	RF
2017	Bahnsen et al.	2 Million URLs (PhishTank, CommonCrawl	LSTM	LSTM
2017	Bhattacharjee et al.	PhishMonger's Targeted Brand Dataset	Ridge Classifier, LR	Mixed results
2017	Li & Wang	47,278 URLs (Alexa, PhishTank)	BN, LR, DT, RF	RF
2016	Mamun et al.	114,400 URLs (Alexa, Heritrix, OpenPhish, DNS-BH, Zone-H, and a publicly available web spam dataset)	C4.5, KNN, RF	RF
2016	Mašetic et al.	5,000 URLs (DMOZ, Yahoo, PhishTank, Spamscatter)	C4.5	C4.5
2014	Akanbi et al.	9,250 URLs PhishTank, Google	C4.5, LR, KNN, SVM	KNN
2014	Marchal et al.	96,018 URLs (Alexa, PhishTank)	RF, RT, LMT, C4.5, PART, JRip, SVM	RF
2013	Chu et al.	46,145 URLs (Toaboa, Yahoo, hao123.com)	SVM	SVM
2013	Eshete et al.	486,384 URLs (Google blacklist, PhishTank, Alexa Top Sites, Yahoo random URL generation, DMOZ)	DT, RT, RF, NB, BN, SVM, LR	RF
2011	Canali et al.	206,573 URLs (Alexa, Wepawet dataset)	RT, RF, NB, LR, J48, BN	J48

Abbreviations can be found in List of Acronyms (page xx).

The majority of works listed in Table 1 focus on traditional ML, though some include NNs, whereas others focus exclusively on NNs. While some papers admitted concern for drift and decay, none performed testing for drift and decay. Most authors admit that making a good dataset can be a challenge when modeling for malicious URL classification, usually for some combination of the reasons mentioned in section 2.5.

Works in Table 1 source their data from a combination of OSINT sources, private sources, and public sources. In the case of OSINT, sites such as PhishTank, OpenPhish,

and URLHaus are exceedingly common for sourcing malicious URL samples, aiding more than half of listed works. Each of these three sites provide the date and time for which malicious URLs were reported, so it is an interesting observation that no research has been found where a temporal analysis was conducted. OSINT sources could be leveraged in producing a temporal URL dataset. Likewise, some research obtained private data from the perimeter proxy logs of universities or colleges that their research was conducted through, which also could have been used in a temporal analysis, as proxy logs would be time stamped. Other private options exist, though, as was the case for Crişan et al. (2020), who sourced their data from an unnamed Romanian cybersecurity company. There are also numerous options for sourcing data from publicly available datasets, such as Kaggle, Alexa Web Crawls, Majestic Million, and Git-Hub to name only a few; however, these datasets generally do not come with time stamps, which may complicate temporal analysis, if used. An alternative option mentioned by some was to use a honeypot. A honeypot is a decoy information system, designed to attract traffic from bad actors so that the honeypot owner can learn from bad actor activities (Bodnar, 2022). Other research reused prior datasets created by other researchers in Table 1.

A further commonality in research is how the validation of dataset labels were handled. Some works neglected or skipped validation, trusting the labels of datasets *in situ*. Most works typically performed validation on a subset of their URLs, with the GoogleSafeBrowsing API and Virus Total API being the most common choices.

2.6.2 Concept Drift Detection Techniques in URL Classification

A few works of research are worth special attention pertaining to Concept Drift in Malicious URL datasets. HEOM (Singhal et al., 2020; Vyawhare et al., 2022) and RSDD

(Tan et al., 2018) are techniques for detecting drift, written in their respective works as either a math equation and/or abstract algebra. These works also evaluate ML models for URL classification; however, they stop short of performing temporal analysis of performance over time and use traditional 70-30 splits in their datasets. The purpose of evaluating models was solely to demonstrate that their datasets perform well enough. Otherwise, a concern for these works would have been that the concept drift detection methods used were tested against useless datasets. Without a temporal analysis, each of these works are left without knowing (a) when concept drift was detected or suspected to occur, in contrast to (b) how performance of the model has changed. Compensating for this, the authors all suggest updating ML models upon drift detection; however, this is only useful for security solutions that actively evaluate their dataset for presence of drift. As such, the research questions of this praxis are not fully addressed by prior research.

Chaitanya R. Vyawhare et al. demonstrates one method for detecting concept drift in a malicious URL dataset in their 2022 work, *Machine Learning System for Malicious Website Detection Using Concept Drift Detection*. In that work the Heterogeneous Euclidean-Overlap Metric (HEOM) is used. HEOM is a formula "for calculating the smallest amount of distance between 2 feature vectors, e.g. malicious old vector and malicious new vector. … If the average of all the smallest amount of distances are calculated for now and is greater than … a pre-set threshold, then the concept drift has occurred" (Vyawhare et al., 2022, p. 53). While the pre-set threshold can be chosen, and could be a bit subjective in selection, HEOM can work with both categorical and numerical attributes, meaning it can handle both lexical and HTML / JavaScript based features. HEOM was also used in 2020 in *Machine Learning & Concept Drift Based*

Approach for Malicious Website Detection (Singhal et al., 2020), with the only significant difference being that fewer features were evaluated in comparison to Vyawhare's work.

The RankSumDriftDetect (RSDD) algorithm is presented in *Adaptive Malicious URL Detection: Learning in the Presence of Concept Drifts* (Tan et al., 2018, p. 740) and is based on incorporating the Wilcoxon Rank-Sum Test (WRST) with a cumulative sum of feature difference measurements. WRST "is a nonparametric test that can be used to determine whether two independent samples were selected from populations having the same distribution" (Tan et al., 2018, p. 740). Tan et al. state that not incorporating the WRST would mean that the cumulative sum "requires that the probability density function and the values of parameters" are known "before and after the concept drift are supposed to be known, which is a quite unrealistic assumption for practical applications" (Tan et al., 2018, p. 740). Similar to HEOM, a predefined threshold is selected, which could be subjective. The difference is that RSDD looks at distribution changes, while WRST looks at inter feature distance changes, and not distribution changes.

Chapter 3 - Methodology

3.1 Dataset Construction

No pre-existing temporal URL datasets meant for classification were identified over the course of this research, meaning a new dataset needed to be created for this praxis. Figure 6 has been included as a reference to help visually describe the dataset's temporal segmentation; this section will review the dataset construction and composition. Data from on or before 1/28/2023 was labelled for training, with data sourced after that labelled into week-long test intervals / buckets. Month-long test intervals were foregone since there was concern that too few test buckets could be produced over the span of this research. Day-long test intervals were foregone since weekends had lower volumes of URLs compared to weekdays. In total, 16 test buckets were created over the 16 weeks between 1/29/2023 and 5/20/2023.

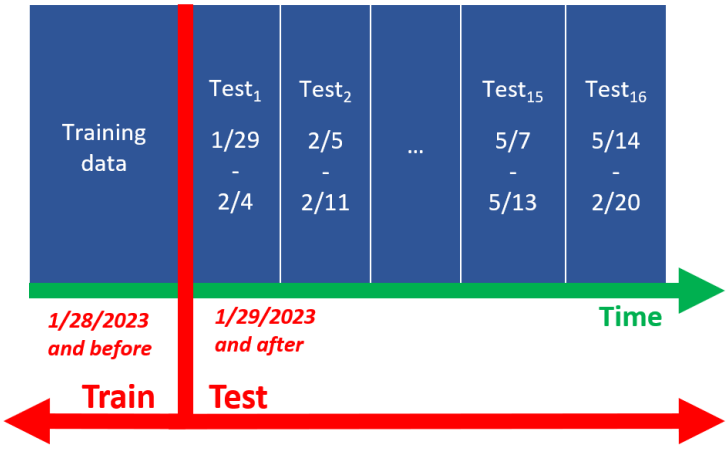


Figure 6: Stratification of Temporal Malicious URL Dataset

Temporal malicious URL datasets require inclusion of a date-time or equivalent feature for each URL's usage so that each URL can be stratified into time-segmented testing buckets. Malicious URLs created prior to 1/28/2023 were eligible for inclusion

within the training portion of temporal datasets, since those URLs would have come from the past and would be known as of 1/28/2023. A nearly universal heuristic is that larger datasets are preferrable to smaller datasets, and for that reason, multiple datasets prior to 1/28/2023 were combined to comprise the training portion of the temporal dataset.

For each dataset sourced, the labels from each source were consolidated to either malicious as '1', or benign as '0'. The malicious '1' label included undesirable labels, such as 'phishing', 'bad', or 'malicious', while the benign '0' label included everything else, such as 'safe' and 'benign'. For each URL sourced, they were first checked for being unique with respect to their malicious or benign label. This was done to avoid duplicates causing possible bias or overfitting, since some datasets contained the same URLs. Second, once all URLs were in one dataset, each URL was checked without respect to their label, and dropped if counted more than once. This was done to remove possible problems where one dataset counted a URL as not safe, where another counted a URL as safe. Third, malformed URL records were dropped after that, too. Table 2 outlines each dataset sourced and Table 3 sums the total of URLs from each dataset sourced.

Table 2: Sources of Data

Source	Citation	Dataset Sources / Description
A	(Wayback Machine, n.d.)	An archive of hosts-file.net; a historic list of URLs that should not be accepted inbound, labeled as malicious.
B	(Marchal et al., 2014)	Alexa, PhishTank
C	(Xuan et al., 2020)	Includes Malicious_n_Non-Malicious URL from Kaggle
D	(Hannousse et al., 2021)	PhishTank, URLhas, Alexa, Malicious_n_Non-Malicious URL
E	(Ghaleb et al., 2022)	Souring from PhishTank and ISCX-URL2016
F	(*URLhaus - API*, n.d.)	CTI of Public reporting of malicious URLs
G	(A. K. Singh, 2020)	Sourced from random web crawls from 11/2019 to 3/2020.
H	(Tyagi, 2023)	A Kaggle dataset released 1/2023
I	(Mitra & Raja M.V, 2021)	OpenPhish, MBPI, malware domain dataset, WEBSPAM-UK2007, ISCX-URL2016, (Mamun et al., 2016)

Table 3: Dataset Counts by Source

Bucket	Label	Total	Source								
			A	B	C	D	E	F	G	H	I
train	1	454,144	45,733	2	66,332	5,712	162,259	126,542	31,691	3,954	11,919
train	0	1,725,631	0	92	344,791	5,714	35,380	0	1,339,654	0	0
$test_1$	1	9,282	0	0	0	0	0	9,282	0	0	0
$test_1$	0	0	0	0	0	0	0	0	0	0	0
$test_2$	1	6,887	0	0	0	0	0	6,887	0	0	0
$test_2$	0	0	0	0	0	0	0	0	0	0	0
$test_3$	1	6,939	0	0	0	0	0	6,939	0	0	0
$test_3$	0	0	0	0	0	0	0	0	0	0	0
$test_4$	1	3,643	0	0	0	0	0	3,643	0	0	0
$test_4$	0	0	0	0	0	0	0	0	0	0	0
$test_5$	1	7,789	0	0	0	0	0	7,789	0	0	0
$test_5$	0	0	0	0	0	0	0	0	0	0	0
$test_6$	1	8,327	0	0	0	0	0	8,327	0	0	0
$test_6$	0	0	0	0	0	0	0	0	0	0	0
$test_7$	1	9,919	0	0	0	0	0	9,919	0	0	0
$test_7$	0	0	0	0	0	0	0	0	0	0	0
$test_8$	1	8,618	0	0	0	0	0	8,618	0	0	0
$test_8$	0	0	0	0	0	0	0	0	0	0	0
$test_9$	1	8,350	0	0	0	0	0	8,350	0	0	0
$test_9$	0	0	0	0	0	0	0	0	0	0	0
$test_{10}$	1	8,731	0	0	0	0	0	8,731	0	0	0
$test_{10}$	0	0	0	0	0	0	0	0	0	0	0
$test_{11}$	1	7,181	0	0	0	0	0	7,181	0	0	0
$test_{11}$	0	0	0	0	0	0	0	0	0	0	0
$test_{12}$	1	5,858	0	0	0	0	0	5,858	0	0	0
$test_{12}$	0	0	0	0	0	0	0	0	0	0	0
$test_{13}$	1	4,819	0	0	0	0	0	4,819	0	0	0
$test_{13}$	0	0	0	0	0	0	0	0	0	0	0
$test_{14}$	1	5,335	0	0	0	0	0	5,335	0	0	0
$test_{14}$	0	0	0	0	0	0	0	0	0	0	0
$test_{15}$	1	5,213	0	0	0	0	0	5,213	0	0	0
$test_{15}$	0	0	0	0	0	0	0	0	0	0	0
$test_{16}$	1	6,216	0	0	0	0	0	6,216	0	0	0
$test_{16}$	0	0	0	0	0	0	0	0	0	0	0

Sources, for the most part, are directly from (or were used in) prior works addressed in Table 1 in Literature Review section 2.6.1. The samples of this dataset collectively cross-sample various sourcing techniques and were selected for the dataset of

this praxis to avoid possible bias from too few sources. Consequently, 567,251 malicious URLs and 1,725,631 benign URLs, summing to 2,292,882 in total, makes the temporal dataset of this praxis one of the largest malicious URL research datasets to date. The dataset is available upon request.

URLHaus, or source F, is a publicly available CTI website that aggregates malicious URLs by the date-time that they were reported. These reported date-times allowed for this temporal analysis to work, as they enabled the malicious URLs to be stratified by the date-time that they were first observed. However, with URLHaus only sourcing malicious URLs, there are no benign samples that occur naturally across the test buckets. As noted in section 1.9, a work-around for this shortcoming is reviewed in section 3.2.2.

3.2 Model Evaluation Considerations

Machine Learning research often scores the performance of models using a combination of Confusion Matrixes, Accuracy, f1-score, Precision, Recall, Receiver Operating Characteristic (ROC) curve, and the Area Under the Curve (AUC), among other techniques (Wood, 2020). In a temporal analysis, the objective is to plot applicable performance metrics across time, or in the case of this praxis, across the 16 test buckets of the URL dataset.

3.2.1 Traditional Performance Metrics

- Confusion Matrix (CM): CM is a technique used for displaying the comparison of predicted versus actual labels, comprising a 2x2 matrix:

$$\text{CM} = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \quad (1)$$

  - True Positive (TP): count of malicious URLs correctly predicted as malicious.

- True Negative (TN): count of benign URLs correctly predicted as benign.
- False Positive (FP): count of benign URLs incorrectly predicted as malicious.
- False Negative (FN): count of malicious URLs incorrectly predicted as benign.

- Accuracy: Accuracy is the proportion of correct predictions over the total number of predictions:

$$\text{Accuracy} = \frac{TP + TN}{TN + TP + FP + FN} \quad (2)$$

- Precision: Precision is the proportion of TP over TP and FP:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3)$$

- Recall: Recall, also known as the True Positive Rate (TPR) or 'sensitivity', is the proportion of TP over TP and FN:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (4)$$

- f1-Score: f1-Score is the harmonic mean of the precision and recall:

$$\text{f1-Score} = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 \text{ x } \frac{recall\ x\ precision}{recall + precision} \quad (5)$$

- ROC: ROC "plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives" (*Classification: ROC curve and AUC*, 2021). A sample ROC graph can be found in Figure 7, below.

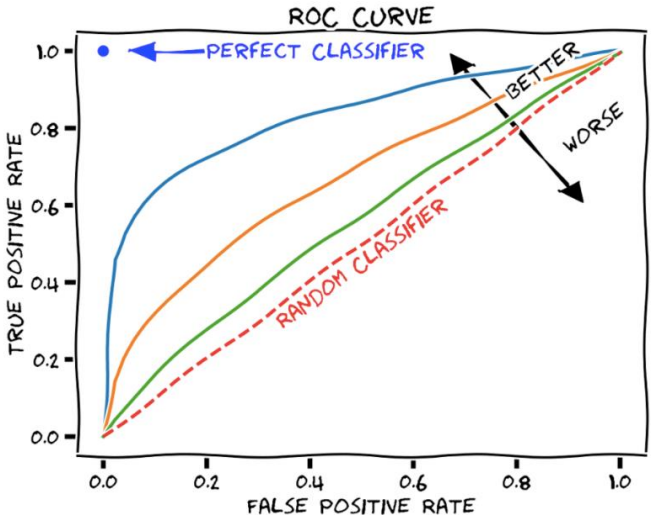


Figure 7: Sample ROC Graph (Kumar, 2023)

- AUC: AUC measures the volume underneath the ROC curve.

3.2.2 Work-Around for Missing Benign Test Data

As noted earlier, no benign samples exist naturally across the test buckets of the dataset of this praxis; however, there are solutions for this shortcoming. This praxis is interested in a temporal analysis of malicious URLs, not benign URLs. Thankfully, malicious URLs can only be classified as either TP or FN. Since recall is entirely calculated using TP and FN, this means recall is a preferred metric for evaluating performance across time.

Further, because benign URLs are not of primary interest to evaluating performance decay, and since we are interested in the performance decay of detecting malicious URLs (the TP and FN instances), a random sample of URLs were migrated from the train bucket to the test bucket. 160,000 URLs were moved in total, with 10,000 benign URLs placed into each test bucket, and 1,565,631 benign URLs left in the training bucket. This at least enables the calculation of all the performance metrics mentioned in section 3.2.1.

Note that it is malpractice to select features based on testing data; however, this migration of URLs from train to $test_{1\text{-}16}$ took place before feature selection was performed, meaning feature selection was not exposed to test data. The primary concern of migrating the benign URLs is that the benign URLs were not stratified by their date of use, meaning they have little to no value for evaluating performance decay. Since all metrics except recall are affected by this stipulation, and since benign URLs can only be TN and FP, all other metrics should take secondary priority to recall, if considered at all. To that point, the Results chapter will evaluate recall and Appendix A will contain results for accuracy, F1, precision, and AUC.

3.2.3 Temporal Considerations

Each Recall, Accuracy, F1, Precision, and AUC need to be plotted to perform a temporal analysis. In order to measure plots, rather than just being something to observe, a line of best fit can be calculated. The objective of the line of best fit is to calculate a single line that will minimize the sum of residual offsets between all points and the line. A visual example for a line of best fit can be seen in Figure 8, below.

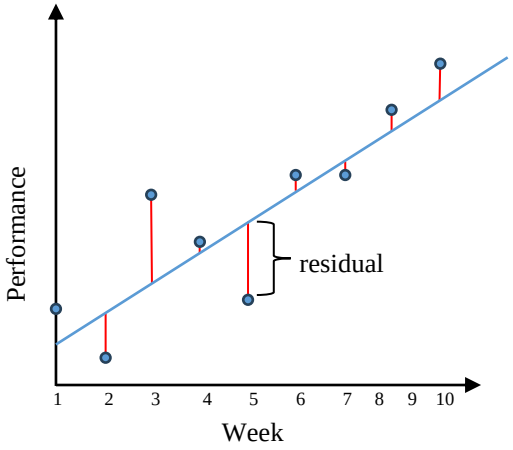


Figure 8: Arbitrary Line of Best Fit

To determine how well a line of best fit predicts what values should be, mean squared error (MSE) can be used (Frost, 2023b). MSE can be calculated by summing the square of each residual and dividing by the total number of residuals. The smaller the MSE is, the better the line of best fit predicts what values should be. A perfect fit would have an MSE of zero, and the purpose of squaring the residuals is to penalize larger errors. In terms of algebra, the formula for MSE is below.

$$\text{MSE} = \frac{\sum(y_i - \hat{y}_i)^2}{n} \tag{6}$$

3.3 Feature Extraction

Prior to Feature Extraction, the dataset was comprised of five columns: 'url', 'label', 'source', 'bucket', and 'dateadded'. Excel was used to expand the dataset to a total of 73 columns with 68 independent variables outlined in Table 4. Of the 68 independent variables, many were also used in prior works, noted in section 2.6.1 (Aljabri et al., 2022; Hannousse et al., 2021; Swarnkar et al., 2022; Xuan et al., 2020). An Excel document used for producing these features is available upon request alongside the dataset.

Table 4: Temporal URL Dataset Columns

Column	Data Type	Description	Comments	Security Intuition
url	string	URL of the row.		
label	integer	label malicious (1) & benign (0).		
source	string	Represents the source of the URL, using the same letter abbreviation in table 3.	This feature should not be used during training or testing, as the model would just assert that every URL from source F is malicious. It exists in the event a data source needs to be removed retrospectively. This feature was not considered during feature selection.	N.A.

bucket	string	Denotes the buckets train, and test1-16.		
dateadded	string	Records from URLhaus (source F) came with the date that they were reported, captured here. Sources A, B, C, D, E, G, H, and I in train were given the value of "0", as well as the benign url records migrated to test1-16.	This feature can be reused to create an alternative timewise segmentation. This feature was not considered during feature selection.	
scheme_https	integer	Binary check of 1 if the scheme of the url used https, and 0 if not.		1.a - By heuristic, unsecure and illegitimate websites do not use https, suggesting https could have a negative correlation to an illegitimate destination.
scheme_http	integer	Binary check of 1 if the scheme of the url used http, and 0 if not.		1.b - Alternate to intuition 1.a, usage of http could have a positive correlation to an illegitimate destination.
scheme_ftp	integer	Binary check of 1 if the scheme of the url used ftp, and 0 if not.		1.c - Alternate to intuition 1.a, usage of ftp could have a positive correlation to maliciousness, as ftp could be used to deliver malware.
scheme_is_ provisional	integer	Binary check of 1 if the scheme is provisional, and 0 if not, based on IANA (Uniform Resource Identifier (URI) Schemes, n.d.).		1.d - Alternative to 1.a through 1.c, this check is included to cover remaining types of schemes.
scheme_is_ permanent _and_not_ http_ https_or_ftp	integer	Binary check of 1 if the scheme is permanent and not http, https, or fpt, and 0 if not, based on IANA (Uniform Resource Identifier (URI) Schemes, n.d.).		
scheme_is_ historic	integer	Binary check of 1 if the scheme is historic, and 0 if not, based on IANA (Uniform Resource Identifier (URI) Schemes, n.d.).		
scheme_is_ undeclared	integer	Binary check of 1 if the scheme is undeclared, and 0 if not.		
scheme_is_other	integer	Binary check of 1 if the scheme is neither historic, provisional, nor permanent, and 0 if not.		
host_is_ip4	integer	Binary check of 1 if host uses ipv4, and 0 if not.		2.a - Usage of an IP address can be used to obfuscate a website's destination and is considered an indicator for an illegitimate destination.
host_is_ip6	integer	Binary check of 1 if host uses ipv6, and 0 if not.		
port_defined_and _ 1024_or_more	integer	Binary check of 1 if host declares a port number of 1024 or higher, and 0 if not.		3.a - Legitimate websites need not rely on use of ephemeral ports, which suggests that use of ephemeral ports indicates an illegitimate destination.
port_defined_and _ 1023_or_less	integer	Binary check of 1 if host declares a port number of 1023 or lower, and 0 if not.		3.b - Legitimate websites need not rely on declaring well-known ports, although the presence of ports such as 443 are less concerning than ephemeral ports. Usage of well-known ports could indicate an illegitimate destination, although to a lesser degree than intuition 3.a.
port_defined_but_ not_number	integer	Binary check of 1 if host declares a non-numeric port, and 0 if not.		3.c - Declaring non-numeric ports is possible in URLs but considered to be abnormal. If

				normal websites use numeric ports, non-numeric ports being used suggests an illegitimate destination.
count_comma	integer	Count of ,		4.a - As a rule of thumb, since simpler urls are generally safer, usage of excessive special characters could indicate an illegitimate destination.
count_dollar	integer	Count of $		
count_ semicolumn	integer	Count of ;		
count_space	integer	Count of single white space		
count_and	integer	Count of &		
count_bslash	integer	Count of \		
count_fslash	integer	Count of /		
count_equal	integer	Count of =		
count_ percentage	integer	Count of %		
count_question	integer	Count of ?		
count_ underscore	integer	Count of _		
count_hyphens	integer	Count of -		
count_dots	integer	Count of .		
count_colon	integer	Count of :		
count_star	integer	Count of *		
count_orbar	integer	Count of |		
count_tilda	integer	Count of ~		
count_open_ parentheses	integer	Count of (		
count_close_ parentheses	integer	Count of)		
count_space_ or_%20	integer	Count of " " or %20		
count_double_ fslash	integer	Count of //		4.b - Double forward slash should only occur, at most, once in a legitimate URL, in addition to intuition 4.a
count_https	integer	Count of https		4.c - Expanding on intuitions 1.a and 4.b, https should only be seen, at most, once in a legitimate URL.
count_http	integer	Count of http	Calculates the count of http and subtracts counts of https.	4.d - Likewise to 4.c, https should only be seen, at most, once in a legitimate URL.
count_dot_com	integer	Count of .com		4.e - Likewise to 4.b through 4.d, .com should only occur, at most, once in a legitimate URL.
count_www_dot	integer	Count of www.		4.f - Likewise to 4.b through 4.e, www. should only occur, at most, once in a legitimate URL.
chars_in_url	integer	Counts the number of letter characters.		4.g - Likewise to intuition 4.a and intuition set 5, more characters could correlate to an illegitimate destination.
digits_in_url	integer	Counts the number of numeric characters.		4.h - Likewise to intuition 4.a and 2.a, excessive numbers could correlate to an illegitimate destination.
count_non_digits_ and_non_chars	integer	Counts all special characters.		4.i - Combines all intuitions denoted as 4.a.
len_authority	integer	Length of the URL's authority.		5.a - Longer URLs are suspected to have a positive correlation to an illegitimate destination, with this feature serving to capture that intuition while also encoding the positional information of the authority.
len_post_ authority	integer	Length of the URL after the authority.		5.b - See 5.b, but for post-authority length.
len_anchor	integer	Length of the URL's anchor.		5.c - See 5.a, but for anchor length.
len_params	integer	Length of the URL's parameters.		5.d - See 5.a, but for length of parameters.

len_path	integer	Length of the URL's path.		5.e - See 5.a, but for path length.
commons_more_ than_once	integer	Binary check of 1 if any of the following occur more than once: www., .com, http, https, //		6.a - Combines the intuitions of 4.b through 4.f.
authority_ends_ in_dot_com	integer	Binary check of 1 if the authority ends in .com, and 0 if not		6.b - .com should only occur at the end of the authority, seeks to encode that positional information, in addition to intuition from 4.e, and could correlate to a legitimate destination.
authority_starts_ www_dot	integer	Binary check of 1 if the authority starts with www., and 1 if not		6.c - www. should only occur at the start of the authority, seeks to encode that positional information, in addition to intuition from 4.f, and could correlate to being benign.
ratio_chars	decimal	Count of letter characters divided by URL length		7.a - Seeks to encode 4.g, relative to total URL length.
ratio_digits_ in_url	decimal	Count of numeric characters divided by URL length		7.b - Seeks to encode 4.h, relative to total URL length.
ratio_non_digits_ and_non_chars	decimal	Count of special characters divided by URL length		7.c - Seeks to encode 4.i, relative to total URL length.
ratio_authority	decimal	Length of ratio divided by length of URL		7.d - Seeks to encode 4.a, relative to total URL length
ratio_anchor	decimal	Length of anchor divided by anchor		7.e - Seeks to encode 5.c, relative to total URL length
ratio_params	decimal	Length of parameters divided by anchor		7.f - Seeks to encode 5.d, relative to total URL length
ratio_path	decimal	Length of path divided by path		7.g - Seeks to encode 5.e, relative to total URL length
count_words	integer	Counts the number of numeric characters		8.a - Expanding on intuition sets 4 and 5, this feature seeks to encode the distribution of special characters to non-special characters.
len_longest_ word	integer	Counts the number of strings separated by any special character, %20, or ://		8.b - Considering the rule of thumb that simpler URLs could indicate a legitimate destination, a high value for this feature could indicate an illegitimate destination.
average_word_ len	decimal	The average length of strings separated by any special character, %20, or ://		
median_word_ len	decimal	The median length of strings separated by any special character, %20, or ://		
absolute_skew_ word_len	decimal	Equal to the absolute value of the prior feature		
count_ parameters	integer	Counts the number of parameters in the URL		
average_ parameter_ len	integer	Equal to the average parameter length		
longest_param_ len	decimal	Equal to the longest parameter length		
skew_word_len	decimal	Is the average length of strings minus the median string length		8.c - This served as an alternate to absolute_skew_word_len
sensitive_ destination_ page_check	integer	Binary check of 1 if any of the following strings are in the URL, and 0 if not: config.bin, dbsys.php, download.php, getImage.asp, login.asp, login.php, login.htm, mail.php, viewer.php		9.a - Seeks to capture terms suspected to correlate to suspicious or malicious destinations, according to (Xuan et al., 2020, p.150; Swarnkar et al., 2022, p. 209; Hannousse et al., 2021, p. 7).
sensitive_file_ extension_check	integer	Binary check of 1 if any of the following strings are in the URL, and 0 if not: .bin, .cgi, .exe, .gif, .jar, .jpg, .js, .rar, .swf, .zip		

sensitive_word_check	integer	Binary check of 1 if any of the following strings are in the URL, and 0 if not: abuse, account, admin, banking, blog, confirm, ebayisapi, files, link=, login, log in, order, payment, paypal, personal, plugins, secure, signin, sign in, signon, sign on, shipping, update, verification, webscr		
random	integer	This is a random integer between 1 and 10,000,000. The feature was used for migrating URLs from train to test.	Was not considered in feature selection.	N.A.

3.4 Feature Selection

Having too many features can have a noticeable impact on how long it takes for Machine Learning algorithms to produce models, as well as how long it takes for those models to produce predictions. As such, a subset of the most useful features is selected, known as feature selection. The most useful features are selected for inclusion by examining the training data and there are common methods used for consistently doing this selection. The two primary techniques used in this praxis for reducing features are variance and correlation.

A feature's variance describes how the feature is spread in comparison to the average value of the feature. Low variance suggests a feature would be a poor predicter of label, and zero variance suggests all records have the same value. Features of lowest or no variance should be dropped. Both scheme_is_historic and host_is_ipv6 were dropped for having zero variance, which occurred because there were no URLs that used ipv6 or historic schemes.

Correlation can be used to compare each feature to one another and to the target label. Features that are highly correlated to each other suggest that the features overlap for the same feature space, suggesting one of the two features could be dropped; the

feature to be dropped is usually the feature with the weakest correlation to the target label. Features with the strongest correlation to the target label are desirable for inclusion.

Figure 9 summarizes a correlation matrix and variance of features, after feature selection reduced the dataset down from 68 independent variables to 21 independent variables. As feature selection can be subjective, these 21 features were selected to strike a balance between valuing their variance and their correlation to both the target label and each other.

Figure 9: Correlation Matrix and Variance of Selected Features

Feature	Variance	x	0	1	2	7	9	11	27	28	39	40	41	44	48	49	53	55	61	62	63	65	67	68
label		0	1.00	0.29	-0.20	0.09	0.43	0.39	0.30	0.13	0.12	0.17	0.06	0.16	-0.38	0.46	0.24	0.19	-0.25	-0.33	0.25	0.13	0.20	0.15
scheme_https	0.02847352	1	0.29	1.00	-0.30	-0.09	-0.04	-0.04	0.02	0.07	-0.03	0.07	0.04	0.06	0.02	-0.01	0.08	0.05	0.00	-0.17	0.05	0.05	0.46	0.03
scheme_http	0.1878965	2	-0.20	-0.30	1.00	-0.92	0.11	0.12	0.19	0.80	0.09	-0.16	-0.03	0.14	-0.15	-0.08	-0.06	0.00	-0.10	0.60	-0.01	-0.03	-0.14	-0.11
scheme_is_undeclared	0.1721868	7	0.09	-0.09	-0.92	1.00	-0.10	-0.11	-0.21	-0.86	-0.08	0.13	0.01	-0.17	0.15	0.09	0.03	-0.02	0.10	-0.56	-0.01	0.01	-0.04	0.10
host_is_ip4	0.04833946	9	0.43	-0.04	0.11	-0.10	1.00	0.91	0.32	0.44	0.00	-0.08	-0.02	0.04	-0.82	0.79	-0.06	0.03	-0.27	-0.25	-0.05	-0.03	0.00	-0.03
port_defined_and_1024_or_more	0.04157243	11	0.39	-0.04	0.12	-0.11	0.91	1.00	0.30	0.48	0.04	-0.08	-0.02	0.04	-0.78	0.76	-0.06	0.03	-0.25	-0.23	-0.05	-0.03	-0.03	-0.04
count_dots	1.07731	27	0.30	0.02	0.19	-0.21	0.32	0.30	1.00	0.31	0.32	0.18	0.13	0.33	-0.39	0.36	0.16	0.32	-0.36	0.07	0.19	0.12	0.12	0.07
count_colon	0.2621962	28	0.13	0.07	0.80	-0.86	0.44	0.48	0.31	1.00	0.08	-0.06	0.02	0.25	-0.44	0.23	0.07	0.12	-0.20	0.37	0.13	0.03	0.02	-0.09
len_authority	36.79747	39	0.12	-0.03	0.09	-0.08	0.00	0.04	0.32	0.08	1.00	-0.14	0.05	-0.01	0.07	0.01	-0.06	-0.05	0.00	0.12	-0.02	-0.03	-0.01	0.09
len_post_authority	1068.315	40	0.17	0.07	-0.16	0.13	-0.08	-0.08	0.18	-0.06	-0.14	1.00	0.15	0.72	-0.01	0.14	0.57	0.79	-0.03	-0.22	0.53	0.69	0.06	0.20
commons_more_than_once	0.006236777	41	0.06	0.04	-0.03	0.01	-0.02	-0.02	0.13	0.02	0.05	0.15	1.00	0.17	-0.01	0.02	0.13	0.18	0.03	-0.04	0.08	0.15	0.00	0.14
count_non_digits_ and_non_chars	28.22725	44	0.16	0.06	0.14	-0.17	0.04	0.04	0.33	0.25	-0.01	0.72	0.17	1.00	-0.21	0.18	0.49	0.93	-0.12	-0.04	0.53	0.33	0.06	0.12
ratio_chars	0.01835301	48	-0.38	0.02	-0.15	0.15	-0.82	-0.78	-0.39	-0.44	0.07	-0.01	-0.01	-0.21	1.00	-0.92	-0.02	-0.19	0.29	0.20	-0.02	-0.02	0.00	0.04
ratio_digits_in_url	0.01384839	49	0.46	-0.01	-0.08	0.09	0.79	0.76	0.36	0.23	0.01	0.14	0.02	0.18	-0.92	1.00	0.08	0.21	-0.22	-0.38	0.06	0.07	0.02	0.02
ratio_params	0.02078469	53	0.24	0.08	-0.06	0.03	-0.06	-0.06	0.16	0.07	-0.06	0.57	0.13	0.49	-0.02	0.08	1.00	0.54	-0.04	-0.10	0.87	0.55	0.08	0.15
count_words	22.90338	55	0.19	0.05	0.00	-0.02	0.03	0.03	0.32	0.12	-0.05	0.79	0.18	0.93	-0.19	0.21	0.54	1.00	-0.11	-0.14	0.58	0.35	0.06	0.15
authority_ends_in_dot_com	0.2446072	61	-0.25	0.00	-0.10	0.10	-0.27	-0.25	-0.36	-0.20	0.00	-0.03	0.03	-0.12	0.29	-0.22	-0.04	-0.11	1.00	0.06	-0.06	-0.03	-0.02	0.01
authority_starts_www_dot	0.2486606	62	-0.33	-0.17	0.60	-0.56	-0.25	-0.23	0.07	0.37	0.12	-0.22	-0.04	-0.04	0.20	-0.38	-0.10	-0.14	0.06	1.00	-0.05	-0.06	-0.12	-0.11
count_parameters	0.9291952	63	0.25	0.05	-0.01	-0.01	-0.05	-0.05	0.19	0.13	-0.02	0.53	0.08	0.53	-0.02	0.06	0.87	0.58	-0.06	-0.05	1.00	0.35	0.06	0.12
longest_param_len	320.9297	65	0.13	0.05	-0.03	0.01	-0.03	-0.03	0.12	0.03	-0.03	0.69	0.15	0.33	-0.02	0.07	0.55	0.35	-0.03	-0.06	0.35	1.00	0.03	0.13
sensitive_file_extension_check	0.02187584	67	0.20	0.46	-0.14	-0.04	0.00	-0.03	0.12	0.02	-0.01	0.06	0.00	0.06	0.00	0.02	0.08	0.06	-0.02	-0.12	0.06	0.03	1.00	0.00
sensitive_word_check	0.03277065	68	0.15	0.03	-0.11	0.10	-0.03	-0.04	0.07	-0.09	0.09	0.20	0.14	0.12	0.04	0.02	0.15	0.15	0.01	-0.11	0.12	0.13	0.00	1.00

Column / row "x" denotes the original index of the column / feature within the dataset.

3.5 Imbalance

Among the datasets training bucket, there are 567,251 malicious URLs and 1,725,631 benign URLs, giving an approximate ratio of 1:3. According to He and Ma, "There is no agreement, or standard, concerning the exact degree of class imbalance required for a dataset to be considered truly 'imbalanced.' But … ultimately what we care about is how the imbalance impacts learning, and in particular, the ability to learn the rare class" (2013). As will be discussed in Results, the next chapter, adequate performance

was reached without the use of balancing solutions. As such, since the rare class was able to be learned, imbalance solutions were foregone.

3.6 Normalization and Standardization

Before performing machine learning, standardization and normalization need to be considered. Standardization rescales each independent variable of "a dataset to have a mean of 0 and a standard deviation of 1", but the values can fall outside of zero and one (Zach, 2021b). Normalization rescales each independent variable of "a dataset so that each value falls between 0 and 1" (Zach, 2021b).

These methods have no effect on variance, nor correlation to the target label, so they do not affect feature selection. What they do affect is how well models can learn to classify the target label, since they ensure that each variable will have a consistent continuous scale. Otherwise, a dataset's independent variables can have a mix of binary, continuous, and discrete variables. The open question in this regard speaks to RQ_4, on what temporal results look like.

3.7 Feature Drift Transformation

Drift among selected features for just malicious URLs were evaluated temporally. The approach taken rescaled features each week to a value between zero and one, since performance metrics are on a scale between zero and one. Only features for malicious URLs were analyzed, since recall is the primary metric of interest, and since recall is based solely on malicious URLs. Transformations in this regard are proportional.

Transforming Features:

$$T_i = \frac{R_i}{Max(R_{1-16})} \tag{7}$$

$$R_i = \frac{S_i}{C_i} \quad (8)$$

R_i denotes the ratio (R) in a given week (i) of S_i over C_i. S_i denotes the sum (S) of all values for the feature within the week, and C denotes the count (C) of all malicious ULRs within the week. Dividing by C is done to account for fluctuation of malicious URLs each week.

T_i denotes the transformed (T) value of each ratio, over the maximum ratio across all 16 test weeks. As Max( ) uses the largest ratio among all 16 test weeks, this scales all ratios into a final transformed value between zero and one. The T_i transformations were then plotted temporally. The malicious URL label count was also transformed to serve as a baseline for comparing features.

Transforming Malicious Label Counts:

$$T_i = \frac{C_i}{Max(C_{1-16})} \quad (9)$$

Likewise to the prior transformation, C_i denotes the count (C) of all malicious ULRs within the week(i), and Max( ) ensures the counts are transformed to a scale between zero and one.

3.8 Algorithm Selection

When it comes to implementing ML, there are many options. In the case of this praxis, algorithms from Scikit-Learn were used. Scikit-Learn is a popular open source library for supervised and unsupervised ML; it is free for public use but has very limited NN methods. The algorithms selected herein were chosen to cross sample the range of ML approaches used in prior works, per Table 1 in section 2.6.1. Summaries of each algorithm used herein are as follows:

- **Logistic Regression (LR)**: From Scikit-Learn unit 1.1.11, LR is a supervised algorithm (*Logistic Regression,* n.d.). In a binary classification analysis, a logistic transformation is applied to the odds of the target's prediction, on a scale of 0 to 1, with one class prediction belonging to prediction probabilities of .5 and lower, and the alternate class belonging to prediction probabilities greater than .5 (Tanner, 2020b). Without the logistic transformation, Logistic Regression would instead be a Linear Regression.
- **Linear Discriminant Analysis (LDA)**: From Scikit-Learn unit 1.2.1, LDA is a supervised algorithm based on dimensionality reduction (*LinearDiscriminantAnalysis*, n.d.). LDA will project independent variables into a lower-dimension space and uses linear discriminants that maximize distance between each class (Tanner, 2021a). These linear discriminants are used in class prediction. Independent variables with stronger separation between each class are given more priority than those with poor separation.
- **Support Vector Machines with a Linear Kernal (SVM-l)**: From Scikit-Learn unit 1.4.6, SVM-l is a supervised algorithm based on Support Vector Machines (SVM) (*Kernel functions*, n.d.). For each feature of a dataset, SVMs will bisect classes with a hype-plane that maximize the distance (also known as margin) between features and features closest to the hyper-plane will influence the resultant decision boundary the most (Bento, 2021). SVM allows a kernel function to be specified, contributing to how the decision boundary is derived. In SVM-l, a linear kernel is used.

- **Support Vector Machines with a Radial Basis Function (SVM-rbf)**: SVM-rbf, likewise to SVM-l, is from Scikit-Learn unit 1.4.6; however, instead of using a linear kernel, a radial basis function (RBF) kernel is used (*Kernel functions*, n.d.). RBF is suitable for classification problems where the difference between classes is non-linear.
- **Linear Support Vector Classification (LSVC)**: From Scikit-Learn unit 1.4.1, LSVC is a supervised algorithm similar to SVM-l, as both use a linear kernel (*LinearSVC*, n.d.). However, while SVM-l uses one-vs-one multiclass reduction, LSVC uses one-vs-rest, resulting in different hyper-planes and different decision boundaries. LSVC is optimized for handling classification in cases where the difference between classes are linear, while SVM-l is not optimized, meaning LSVC should take less time to run than SVM-l when applied to linear problems.
- **Stochastic Gradient Descent (SGD)**: From Scikit-Learn unit 1.5.1, SGD is only suitable for linear classification (*Stochastic gradient descent*, n.d.). SGD determines how to bisect features by iteratively comparing decision boundaries using the gradient descent optimization technique. This learner used the hinge loss configuration, making it similar to LSVM. However, SGD is generally more efficient than LSVM in terms of time, but at the cost of lower performance.
- **K-Nearest Neighbor (KNN)**: From Scikit-Learn unit 1.6.2, KNN is a supervised method that uses proximity to infer class predictions (*KNeighborsClassifier*, n.d.). KNN does not train models. Rather, the entire dataset is stored into memory; for the record to have its class predicted, the nearest k data points are compared and

the majority class among neighbors is used (Tanner, 2020a). Comparison of three neighbors is standard and three neighbors were used in the case of this praxis.

- **Gaussian Naïve Bayes (GNB)**: From Scikit-Learn unit 1.9.1, GNB is a supervised method based on Naïve Bayes (NB) (*Gaussian naive bayes*, n.d.). NB classifiers use Bayes Theorem under rather basic assumptions, hence the name 'naïve' (IBM, n.d.). These assumptions include features being unrelated to each other, and features contributing equally to classification. GNB is suitable for handling continuous variables under normal (or Gaussian) distributions, finding for each class the standard deviation and mean.
- **Bernoulli Naïve Bayes (BNB)**: BNB, like GNB, is from Scikit-Learn unit 1.9.4, but instead of using Gaussian distributions, BNB uses Bernoulli distributions (*Bernoulli naive bayes*, n.d.). BNB is appropriate for handling binary variables instead of continuous variables (IBM, n.d.).
- **Decision Tree Classifier (DTC)**: From Scikit-Learn unit 1.10.1, DTC is a supervised method that builds chains of decisions for predicting class, known as a tree (*Decision trees*, n.d.). A DTC is built by continuously dividing the dataset into smaller samples, and at each division, considering all features on how best to divide the dataset further. This is repeated until a stop condition is met, with Scikit-Learn stopping based on cost-complexity.
- **Random Forest (RF)**: From Scikit-Learn unit 1.11.2, RF is a supervised ensemble method based on a forest of randomized trees, with the ensemble based on Bootstrap aggregating (also known as Bagging) (*Ensembles: gradient boosting, random forests, bagging, voting, stacking*, n.d.). An ensemble denotes

that multiple models are combined. RF makes trees by subdividing the dataset likewise to DTC; however, each split considers a random subset of all features, rather than considering on all features (Tanner, 2021b). Further, multiple random trees are generated and then combined to create a final Random Forest.

- **AdaBoost**: From Scikit-Learn unit 1.11.3, AdaBoost is a supervised method based on ensembling weak, small decision trees, called decision stumps (*AdaBoost*, n.d.). Stumps are generated iteratively, with each additional stump focused on trying to predict classification of samples that previous stumps missed. The final classification is based on the weighted majority vote of all stumps; stumps that can correctly predict difficult-to-classify records are given more weight in their vote.
- **Ensemble Voting Classifier (Ensemble)**: From Scikit-Learn unit 1.11.6, Ensemble combines multiple learners (*Voting classifier*, n.d.). In the case of this praxis, Ensemble was set to have a soft vote, meaning that each ensembled learning had their predictions averaged together (Simic, 2023). This was chosen in place of a majority vote. For determining which models to combine, a 70-30 split was made from a random 1% subsample of training data, and all aforementioned models were evaluated for performance. This was done as a quick approach for determining which algorithms would likely produce well performing models. By that method, in the case of this praxis, RF, DT, KNN, and LR were selected to be used together in Ensemble.

3.9 Model Generation

Considering computational overhead in terms of time, a 3% fraction of the training dataset was used. That 3% slice was split with 70% used for training and 30% used for validation, so that the learners could have their temporal metrics contrasted with the training performance. This means each model was trained on approximately 9,537 malicious URLs (454,144 x .03 x .7) and 36,238 benign URLs (1,725,631 x .03 x.7).

That process was repeated 60 times in total, with 20 iterations run with the application of either normalization, standardization, or neither. Each iteration trained, validated, and tested each model reviewed in section 3.8. The results of these iterations were grouped by their application of either normalization, standardization, or neither, then averaged, and then plotted.

3.10 Spearman's Rank Correlation Test

A correlation coefficient can be used for measuring association between two variables X and Y, stating how much one variable influences the other. A correlation coefficient can be between any two variables, including features and performance metrics. To this end, knowing correlations can be desirable for evaluating hypotheses.

The Pearson correlation test is a common method for deriving correlation coefficients, but Pearson's assumptions include that variables have no outliers, and that the variables are normally distributed (Zach, 2021c). Nearly half of variables did not meet these assumptions; outliers were confirmed by checking interquartile range outlier fences and lack of normal distribution was confirmed with the Jarque-Bera test (Frost, 2023a; Zach, 2021a).

With Pearson ruled out, Spearman's rank correlation test was used instead. The difference is that the Spearman's correlation test evaluates the "strength and direction of monotonic relationship" while Pearson's correlation test evaluates the "strength and direction of linear relationships" (Sereno, 2023). Spearman's correlation test requires variables to be ordinal for the method to work, meaning numbers must be a rank-ordered set, or numbered one through sixteen in the case of this praxis, since there are 16 test weeks. Once in rank-order, the difference (d_i) between each variable's rank-order was used to create the Spearman's Correlation Coefficient (R_s).

Spearman's Correlation Coefficient

$$R_s = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)} \tag{10}$$

$$d_i = Ranked(X_i) - Ranked(Y_i) \tag{11}$$

The equation for R_s uses *n* to denote the number of variable pairs being compared, which is 16 in the case of this praxis. A correlation coefficient on its own, however, does not state if the correlation is significant. Determining significance of correlation was done with a hypothesis test statistic (t-test) and probability value (p-value). The significance level (α), of .05 was used.

The null hypothesis for Spearman's Correlation Coefficient states that the correlation is not significant, meaning there is no relationship between variables. The alternative hypothesis states that the correlation is significant, meaning there is relationship between variables. If a p-value is less than or equal to the chosen α of .05, the null hypothesis is rejected and the alternative hypothesis is accepted (the correlation is significant). If a p-value is greater than the chosen α of .05, then the null hypothesis cannot be rejected (the correlation is insignificant).

3.10.1 Assumption Criteria for Spearman's Rank-Order Correlation Test

There are five assumptions that must be met for a Spearman correlation test to be valid (*Spearman's Rank-Order Correlation - A Guide to When to Use It, What It Does and What the Assumptions Are*, n.d.; Bradburn, 2020b, 7:20; Sereno, 2023).

1. **Ordinal, Ratio, or Interval**: Variables must be continuous (ratio or interval), or on an ordinal scale. Rank-ordering each variable meets this requirement.
2. **Random Sample**: The sample used should be truly random and be representative of the total population. In this case, the sample refers to the test malicious URLs from URLHaus, and the total population refers to all malicious URLs. It can be argued that the test URLs from URLHaus are representative of all malicious URLs and that the open-source reporting of URLHaus leaves the sample up to the random whimsy of what users independently report, meeting this requirement.
3. **Paired Variables**: For any given variable X_i there is a corresponding variable Y_i. This is to say, there are no missing variables, which is the case.
4. **Independent Observation**: There should be no relationship between the subjects. For example, this means that measuring the X_i and Y_i variables of one URL does not influence the X_i or Y_i variables of another URL, which is the case.
5. **Monotonic Association**: Perhaps the trickiest assumption to meet is that the relationship between variables must be monotonic, meaning "as the value of one variable increases, so does the value of the other variable", or "as the value of one variable increases, the other variable value decreases." Checking for the presence of a monotonic association was confirmed by plotting two variables in a scatter plot. The relevant plots on this check can be found in Appendix B.

3.11 One-Way Analysis of Variance

One-Way Analysis of Variance (ANOVA), as a parametric test, can be used to determine if there is statistically significant difference between the mean values of three or more categories; however, ANOVA will only state as to whether there is a presence of a statistically significant difference (Frost, 2019; Frost, 2021). If ANOVA determines that a statistically significant difference exists between categories, then a post hoc test can be used to determine which categories are different from others. There are various post hoc tests, but the commonality between them is that a post hoc test will group the categories of the ANOVA test. Each category will have an insignificant statistical difference from each other category within the same group. Each category will have significant statistical difference from each category not within the same group.

ANOVA's null hypothesis states that the means of all categories are equal, with ANOVA's alternate hypothesis stating that means are unequal (Zach, 2022). The product of performing ANOVA is a p-value. If ANOVA's p-value is less than an α, then the null hypothesis of equal means can be rejected, and a post hoc test can then be performed. If ANOVA's p-value is not less than α, then the null hypothesis cannot be rejected, concluding that there is no difference between category means, and a post hoc test can be foregone. Looking at the hypotheses of this praxis, ANOVA and post hoc tests were desirable to determine if certain models or scaling techniques are better than others (or not), but some assumptions must be met for the use of ANOVA to be valid.

3.11.1 Assumptions Criteria for One-Way Analysis of Variance

There are six assumptions that must be met for an ANOVA test to be valid (Daniel, 2017, 2:04; Fein, 2022; Hassan, 2023).

1. **Categorical and Random:** The independent variables of the ANOVA test must be categorically (or nominally) separable, with three or more categories of roughly equal size, with samples in each category randomly selected. The dataset was design to be representative of all URLs used on the internet, and with each iteration of training based on a randomized 70% slice of a randomized 3% fraction, it can be argued that randomness was achieved. For categories to be of roughly equal sizes, the count of samples of the largest category, if divided by count of samples of the smallest category, should not exceed 1.5. These assumptions on roughly equal sizes, randomness, and categorical separation were met herein.
2. **Quantitative and Scale:** The dependent variable (e.g. recall) of each sample must be quantitative and on a continuous scale level. As all subjects herein used interval measures (they all fell within a continuous scale), this assumption was met.
3. **Outliers**: Outliers should be removed. ANOVA permits removal of outliers so long as the roughly equal sizes requirement of assumption one holds true after outliers are removed. Outliers were removed if identified by checking interquartile range outlier fences (Frost, 2023a). After outlier removal, the assumption for roughly equal sizes was met between all categories, meeting this assumption for all ANOVAs performed herein.
4. **Normality**: Scores of dependent variables should be normally distributed. After outliers were removed, presence of normal distribution was confirmed for each category using the Jarque-Bera test, meeting this assumption (Zach, 2021a).
5. **Independence**: The scores of a subject in one category should not affect the score of a subject within another category (the same subject shouldn't be in multiple groups).

This assumption, for example, would be violated in a clinical weight loss trial if a patient consumed multiple test drugs and was included as a sample within the category of each drug. No subjects herein were repeated between any categories, meeting this assumption.

6. **Homogeneity of Variance:** This assumption states that the variance of each category should be proportional. This can be determined using Levene's test for equality of variances. The null hypothesis of Levene's test states that all variances are proportional, and the alternate hypothesis of Levene's test states that variances are not proportional. Levene's test produces a p-value. If Levene's p-value is not less than an α (.05 was used in this praxis), then Levene's null hypothesis cannot be rejected, and it can be said that variances are proportional between categories. If Levene's p-value is less than an α, then Levene's null hypothesis is rejected, and it can be said that the variances are not proportional between categories. Every set of data candidate for ANOVA did not meet this assumption, with Levene p-values all being below the α of .05; however, as will be discussed next, there is a solution to not meeting this assumption.

3.11.2 Welch's One-Way Analysis of Variance

Welch's ANOVA is an alternate to ANOVA and while Welch's ANOVA assumes the first five assumptions of ANOVA (see section 3.11.1) to be true, Welch's ANOVA does not assume homogeneity of variance to be true (Frost, 2017). As Levene's test failed for every set of data candidate for ANOVA, Welch's ANOVA was selected for use in place of ANOVA. The null and alternative hypotheses of Welch's ANOVA are identical to ANOVA, and the appropriate post hoc test for Welch's ANOVA is the

Games-Howell grouping method (Schlegel, 2020). Welch's ANOVA and Games-Howell post hoc tests were performed using Minitab.

Chapter 4 - Results

4.1 Introduction

Recognizing considerations from Methodology, evaluating results of 16 test buckets comes with the challenge of not being able to cleanly contain all data within standard 8.5x11 inch paper. As such, summarizations of the results are used herein, with raw results data available upon request.

4.2 Description of Graphs and Tables for Results

Section 4.3.1 compares learners with respect to how long each algorithm's model took to train and test in terms of time. Section 4.3.2 reviews findings from the feature drift transformations outlined in section 3.6. Sections 4.3.3 through 4.3.5 each contain a temporal graph and two tables for recall. Respectively, sections 4.3.3 through 4.3.5 address recall results for data based on whether the dataset was not rescaled (i.e. Non-Rescaled), normalized, or standardized. Sections 4.3.6 and 4.3.7 analyze the tables from sections 4.3.3 through 4.3.5, collectively. Section 4.3.8 reviews a correlation analysis, followed by Welch's ANOVA in 4.3.9 and 4.3.10, and hypothesis discussion in 4.4.

The temporal graphs show performance plotted against the validation portion of data (denoted as '$train_0$') and the 16 test weeks (denoted $test_{1\text{-}16}$). The purpose of including the $train_0$ score within the graphs was to help visualize the difference between how well the model performed during training versus the temporal test buckets. The idea here is that performance during training should be reflective of the kinds of results that a traditional 70-30 split may share. Additionally, graphs share results for all 16 weeks for each model, while the subsequent tables do not.

The first table accompanying each temporal graph shares the rank-ordering change between train, $test_1$, and $test_{16}$. As their titles suggest, the rank-ordering tables provide comparison for determining which models performed the best in contrast between $train_0$, $test_1$, and $test_{16}$. The rank-ordering tables are sorted by performance in $test_{16}$ and help to numerically state what the temporal graphs show. These rank-ordering tables do not demonstrate what occurred between $test_1$ and $test_{16}$, while the temporal graphs do.

The second table accompanying each temporal graph shares the formula for a linear line of best fit and respective MSE. As an alternative to rank-order comparisons, the slope of line of best fit helps to understand the rate at which performance changes in aggregate, and the MSE helps to determine the reliability of the best fit lines. The lines of best fit were calculated using $test_{1\text{-}16,}$ not train, and the tables for lines of best fit are sorted from highest to lowest slope. Note that the linear line of best fit tables show 'm' and 'b' at a 100x scale in comparison to the rank-order tables and temporal graphs. For example, this means that if we have an 'm' of -.05 and 'b' of 99, that would equate to a -.0005 decrease each week, with the $test_1$ intersected at .99. This difference in scale was done to avoid possible issues in how MSE is calculated, as squaring errors smaller than 1 will decrease the mean of MSE.

As mentioned in Methodology, recall is of foremost interest. Accuracy, F1, precision, and AUC have little to no temporal value for the dataset created for this praxis, but are included in Appendix A for the sake of transparency. Likewise, Appendix A also has results for data based on whether the dataset was Non-Rescaled, Normalized, or Standardized.

4.3 Performance Measurements

4.3.1 Time

Table 5 shares the total time that was needed on average for each model to be trained and then tested, measured in seconds. Since there were a total of 60 iterations for each model, with 20 iterations each for normalization, standardization, and non-rescaled approaches, the standard deviation of time (Avg time Stdev) has been included.

Table 5: Time Taken to Train and Test Each Model

Approach	Normalization Speed			Standardization Speed			Non-Rescaled Speed		
Model	Rank	Avg (s)	Stdev (s)	Rank	Avg (s)	Stdev (s)	Rank	Avg (s)	Stdev (s)
GNB	1	0.42344	0.01417	1	0.4828	0.01832	1	0.43454	0.02169
DTC	2	0.5983	0.0189	2	0.66655	0.0231	2	0.62636	0.02853
BNB	3	0.7897	0.14745	3	0.84037	0.09673	3	0.74917	0.0665
LDA	4	0.84514	0.14828	4	0.81859	0.06489	4	0.7581	0.07563
SGD	5	0.84586	0.05929	5	0.75004	0.04517	5	0.84993	0.07751
LR	6	1.01142	0.21402	6	1.03715	0.07751	6	1.11623	0.22607
LSVC	7	1.29845	0.11681	7	7.72989	0.54441	7	4.60978	0.28865
AdaBoost	8	5.66138	0.06989	8	6.36692	0.18716	8	6.21014	0.23773
RF	9	7.58237	0.09049	9	8.51977	0.21164	9	8.35802	0.32283
KNN	10	32.0966	0.49059	10	36.0291	1.10494	10	36.1542	1.66669
Ensemble	11	34.9165	0.75096	11	39.4802	1.16266	11	39.8492	1.96124
SVM-rbf	13	328.435	11.5504	12	319.705	9.50353	12	498.791	13.0779
SVM-l	12	243.479	2.39309	13	370.764	15.0064	13	2336.88	207.251

These times were reached using only 3% of training data and significantly more time was needed for the slowest models. In total, approximately 24 hours was needed for all computation. SVM-rfb and SVM-linear accounted for nearly 93% of all time spent. On its own, non-rescaled SVM-l accounted for nearly 13 hours (2337 seconds / 60 seconds per minute *x* 20 iterations / 60 minutes in an hour), which is slightly more than half of the total run time. By rank-order, the recalling approach made no difference in each case except for the 12th and 13th rank for SVM-rfb and SVM-linear. However, as will be shown next, these two SVM algorithms generally outperformed their competition in performance. These speeds were achieved using Google's Colab Pro, a cloud-based Jupyter environment for programming. Colab dynamically allocates GPUs based on

demand and availability. The specific GPU used for achieving the speeds listed above was not recorded; however, the GPU was most likely either a Nvidia T4, V100, or A100.

4.3.2 Feature Transformation Evaluation

Too many lines within a line plot are unreadable. As such, this section breaks the temporal analysis of the features into 4 separate plots. Features were initially grouped based on their respective category (integers, binary, or decimal), but grouping features in that manner was not insightful. Instead, this section reviews temporal plots of features based on visual similarity. Additionally, each plot includes the T_i of the transformed malicious URL count (the green line in each plot) as a baseline for comparison.

To review an important observation, these feature plots exclude analysis of features from the training portion of the dataset. Training data was excluded because the training URLs spanned several years, while the test buckets are just week-long intervals. Further, plotting with the inclusion of training data was attempted; however, the transformation led to the training bucket having the highest score of one, and nearly all other features condensed to indistinguishable lines near zero, with little to no visual difference.

The feature “scheme_is_undeclared” was also omitted from these temporal plots. Though the analysis of correlation and variance suggested “scheme_is_undeclared” would be a good feature for classifying malicious URLs, all malicious URLs among the 16 test weeks had their scheme declared. That means “scheme_is_undeclared” was a dead variable, with no utility for classifying the malicious URLs.

Examining Figure 10, the feature plot contains URLs with the most apparent direct and indirect correlation to the count of malicious URLs each week. All but

"len_post_authority" have a positive correlation. This reveals that, in weeks with the lowest reporting of malicious URLs (test weeks 4, 13, and 14, and 15), the proportion of URLs with colons, longer authority lengths, and port declarations above 1024 were lower. Conversely, in weeks with lower reporting of malicious URLs, the proportion of URLs with length after the authority were higher.

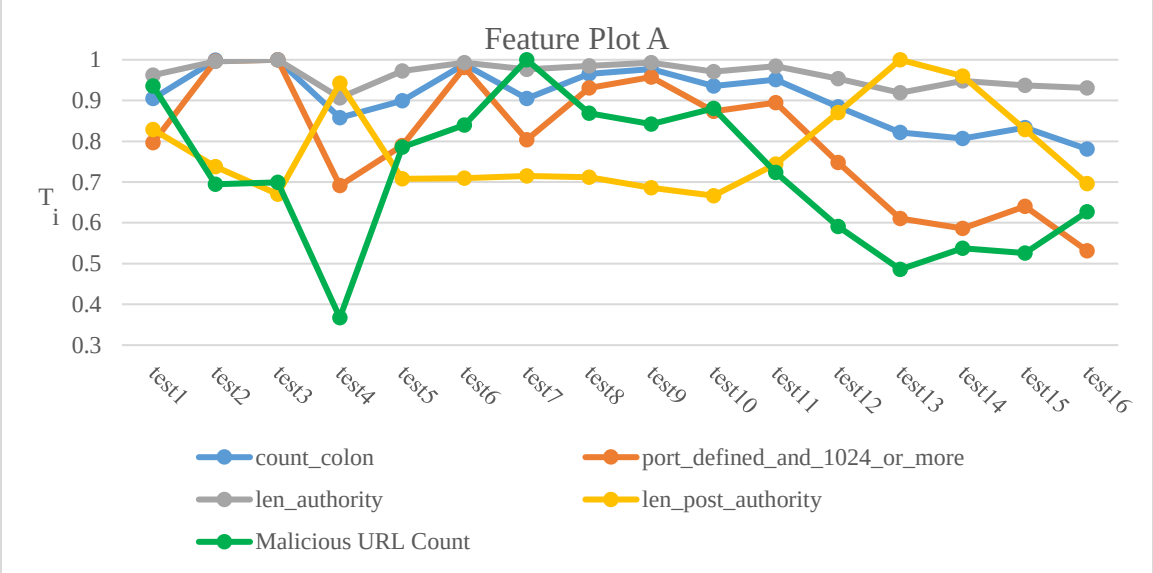


Figure 10: Temporal Feature Plot

Secondary to Figure 10, Figure 11 contains features with visual similarity to one other; however, the features in Figure 11 lack the same low points found in $test_4$ and $test_{13}$. There are fewer insights in Figure 11, but what can be observed is that some features share similar proportions. For example, if there are more ipv4-based URLs in a given week (host_is_ipv4), there will be a higher ratio of digits (ratio_digits_in_url) in that week, and there will be a smaller ratio of characters in that week (ratio_chars). Though these findings are not altogether surprising, they do reflect the correlation analysis in Figure 9 of section 3.4.

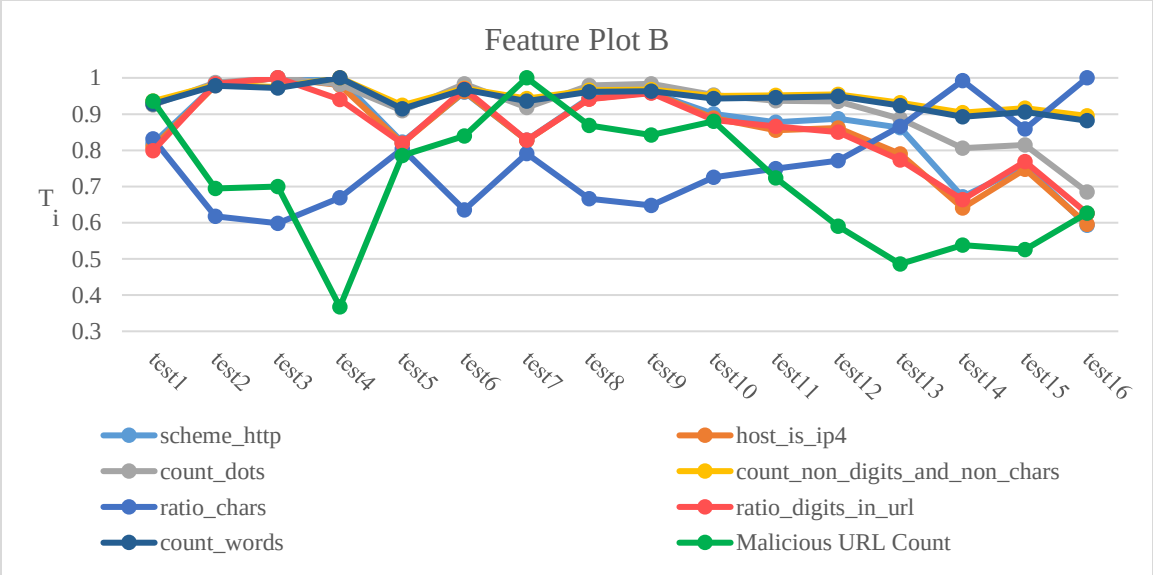


Figure 11: Temporal Feature Plot

Figure 12 includes features in which one week held a disproportionately high T_i, or an outlier of proportion. As a result, the non-outlier weeks of the respective line were closer to zero. A possible concern for these features is their reliability, as ideal features for ML classification are consistent across time. Large swings in proportionality could hurt these features' desirability if models are too reliant on them for classification.

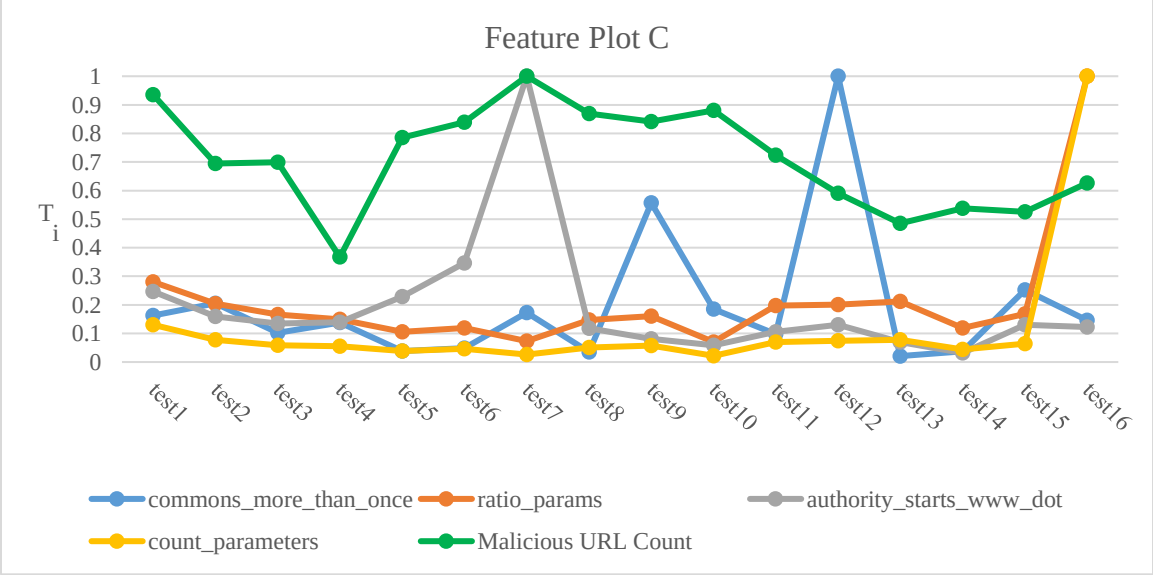


Figure 12: Temporal Feature Plot

Figure 13 contains the remaining features. The overall insight from these four feature plots is that features do indeed fluctuate in proportionality over time with respect to their target label (Malicious URL Count), and none of the selected features were absolutely consistent, but some features were more consistent than others.

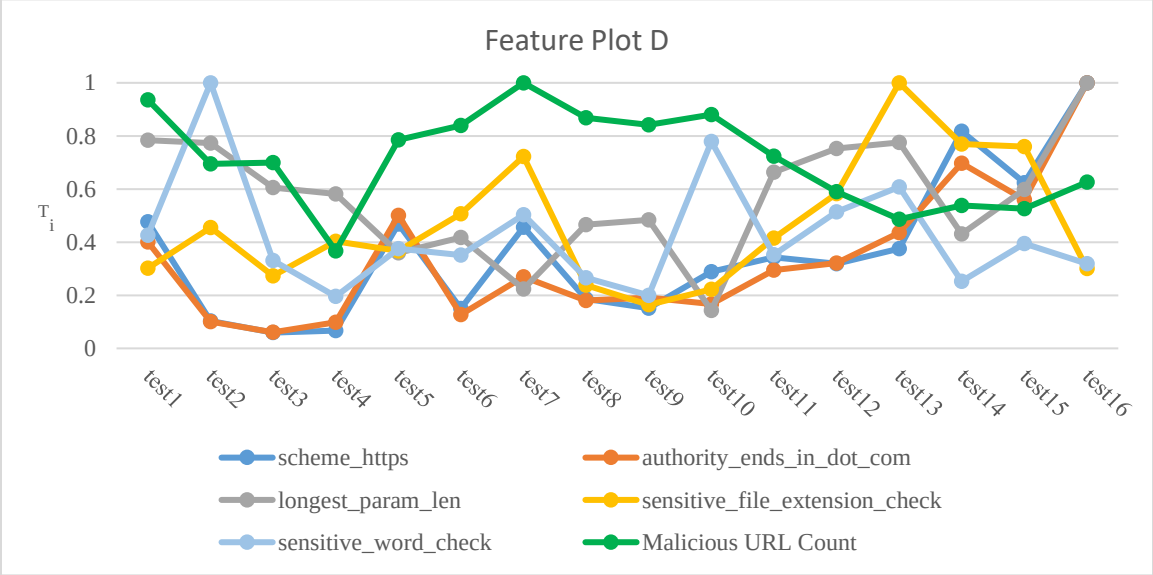


Figure 13: Temporal Feature Plot

An observation of possible concern, which will be observable across sections 4.3.3 through 4.3.5, is that recall of all models seems to correlate to the high and low points of the Malicious URL Counts. This is to say, where there are fewer malicious URLs reported in a given week, model performance also drops, and when there are more malicious URLs reported, performance improves. Further speculation in this regard is reviewed in the correlation analysis of section 4.3.8.

4.3.3 Non-Rescaled Recall

Figure 14 presents the first temporal plot for recall where the dataset was not rescaled. $Train_0$ shows the recall from data on or before 1/28/2023, reflective of what a 70-30 split approach would show. The differences among the types of URLs among the

training data, versus those from URLHaus in $test_{1\text{-}16}$, explains the jump in performance from $train_0$ to $test_1$. Models generally do not change in their rank-ordering across test weeks. Further, all models share roughly the same weekly highs and lows, and $test_4$ and $test_{13}$ consistently show low recall. Performance also seems to mirror the same highs and lows of the Malicious URL Counts from section 4.3.2 where performance drops in weeks during which fewer malicious URLs were reported. As mentioned, the correlation analysis of section 4.3.8 reinforces this observation.

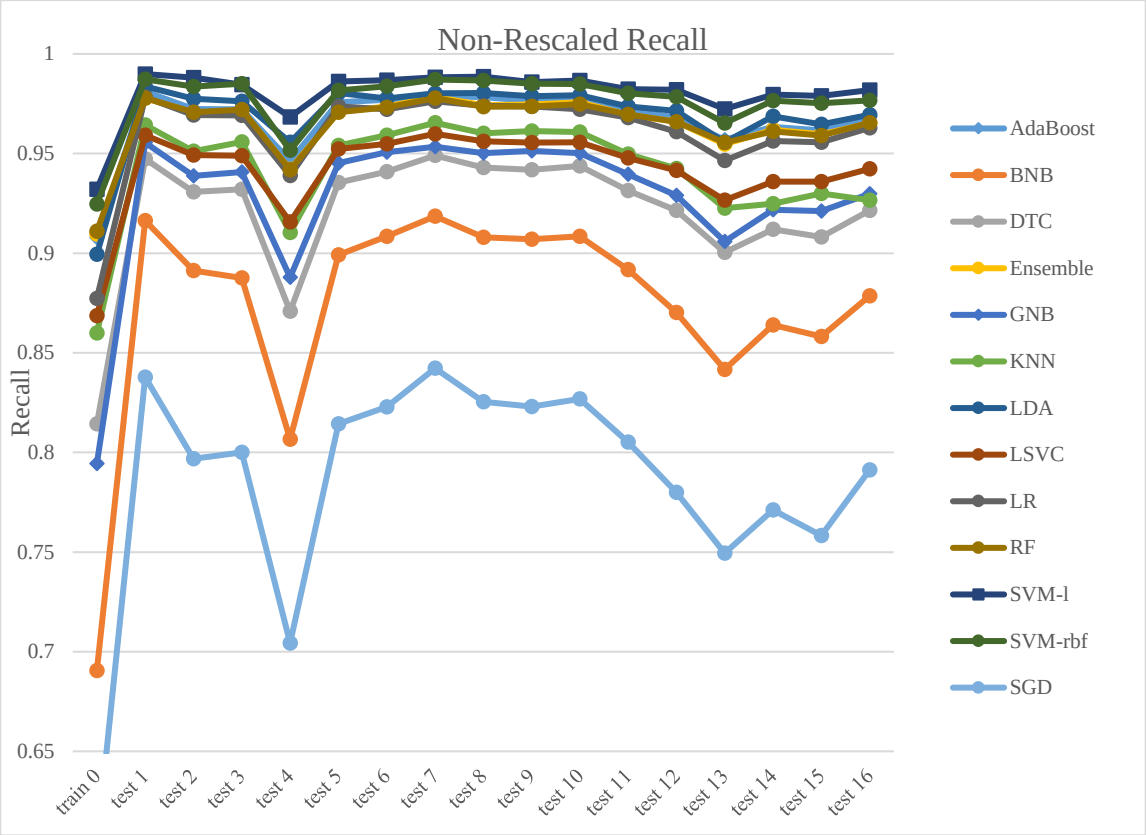


Figure 14: Non-Rescaled Recall Temporal Graph

Looking to Table 6, each non-rescaled model's recall rank-ordering for $train_0$, $test_1$, and $test_{16}$ can be reviewed. Initially, LDA and AdaBoost jumped from ranks six and five to ranks three and four, respectively. A 70-30 split approach selecting models by their recall would not have shown this change of rank. On the other hand, LDA,

AdaBoost, RF, and Ensemble are within a similar margin of recall (an approximate 0.6% maximum difference) in test weeks one and 16.

Table 7 shares the line of best fit for each non-rescaled model's recall. In light of recall seeming to mirror the same highs and lows of the Malicious URL Counts from section 4.3.2, the negative slope for the lines of best fit may be reflecting the fact that there are fewer malicious URLs in the latter end of the 16 weeks. The MSE, on the other hand, does reflect which models have the most linear recall.

Table 6: Non-Rescaled Recall Rank-Order

Model	train Recall	train rank	test1 Recall	test1 rank	test16 Recall	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.93204	1	0.98988	1	0.98183	1	0.05784	-0.00805
SVM-rbf	0.92458	2	0.98724	2	0.97676	2	0.06266	-0.01048
LDA	0.89937	6	0.98367	3	0.96948	3	0.08430	-0.01420
AdaBoost	0.90783	5	0.98111	4	0.96852	4	0.07328	-0.01259
RF	0.91103	3	0.97787	6	0.96544	5	0.06684	-0.01243
Ensemble	0.90949	4	0.97784	7	0.96513	6	0.06835	-0.01271
LR	0.87726	7	0.97825	5	0.96282	7	0.10099	-0.01543
LSVC	0.86859	8	0.95922	9	0.94235	8	0.09063	-0.01687
GNB	0.79435	11	0.95548	10	0.92982	9	0.16113	-0.02566
KNN	0.86002	9	0.96437	8	0.92662	10	0.10435	-0.03775
DTC	0.81427	10	0.94725	11	0.92146	11	0.13298	-0.02579
BNB	0.69059	12	0.91624	12	0.87853	12	0.22565	-0.03771
SGD	0.60286	13	0.83778	13	0.79117	13	0.23493	-0.04661

Table 7: Non-Rescaled Recall Best Fit Line and MSE

Model	m Recall	m rank	b Recall	b rank	MSE Recall	MSE rank
SVM-rbf	-0.04191	1	98.28780	2	0.78574	5
SVM-l	-0.04697	2	98.71208	1	0.28942	1
Ensemble	-0.05687	3	97.27078	6	0.79111	6
RF	-0.05831	4	97.26260	7	0.76671	4
AdaBoost	-0.06138	5	97.56030	4	0.73845	3
LDA	-0.08088	6	98.01952	3	0.55123	2
LSVC	-0.08733	7	95.35129	9	1.28780	8
LR	-0.08858	8	97.30106	5	1.00635	7
DTC	-0.11745	9	93.68382	11	3.75421	11
GNB	-0.12347	10	94.61692	10	3.05450	10
KNN	-0.17849	11	96.13513	8	2.23285	9
BNB	-0.18512	12	90.04325	12	7.99710	12
SGD	-0.21914	13	81.54315	13	11.76663	13

recall = (mx+b)/100

4.3.4 Normalized Recall

Upon initial examination, when comparing Figure 14 to Figure 15, recall from the Normalized dataset has little to no visual difference to recall of the Non-Rescaled dataset. Although sections 4.3.6 and 4.3.7 will review differences between rescaling approaches for recall in further detail, an obvious difference between the plots is that SGD jumped to being a top performer, from the worst. General observations from the Non-Rescaled approach are also true: (a) performance jumps from $train_0$ to $test_1$; (b) rank-orderings generally remain the same in test weeks; and, (c) recall seems to mirror the same highs and lows of the malicious URLs reported each week, which can be seen in the Malicious URL Counts from section 4.3.2.

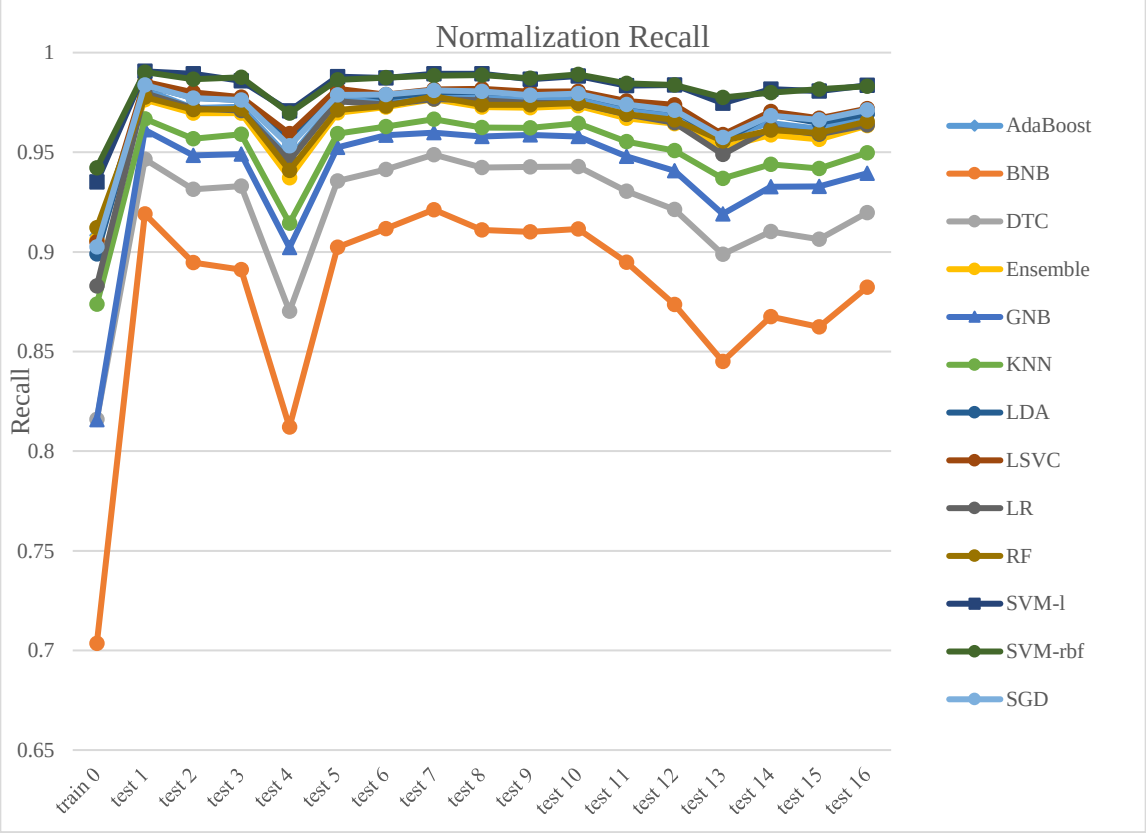


Figure 15: Normalized Recall Temporal Graph

Looking to Table 8, each normalized model's recall rank-ordering among $train_0$, $test_1$, and $test_{16}$ can be reviewed. Looking to the top four performers between $train_0$ and $test_1$, again, a trade in rank can be observed: LSVC and SGD jump from ranks six and seven to ranks three and four, respectively. RF dropped from rank three in $train_0$ down to rank eight in $test_1$, and AdaBoost dropped from rank four in $train_0$ down to rank six in $test_1$. Conversely, LSVC, SGD, RF, and AdaBoost are again within a similar margin of recall (an approximate 0.8% maximum difference) in test weeks one and 16. Table 9 shares the line of best fit for each normalized model's recall.

Table 8: Normalized Recall Rank-Order

Model	train Recall	train rank	test1 Recall	test1 rank	test16 Recall	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.93510	2	0.99059	1	0.98351	1	0.05550	-0.00708
SVM-rbf	0.94224	1	0.99024	2	0.98312	2	0.04800	-0.00712
LSVC	0.90510	6	0.98534	3	0.97190	3	0.08025	-0.01344
SGD	0.90242	7	0.98371	4	0.97125	4	0.08129	-0.01246
LDA	0.89897	8	0.98367	5	0.96932	5	0.08470	-0.01435
AdaBoost	0.90828	4	0.98159	6	0.96857	6	0.07331	-0.01302
RF	0.91216	3	0.97770	8	0.96552	7	0.06553	-0.01218
LR	0.88286	9	0.97970	7	0.96366	8	0.09684	-0.01604
Ensemble	0.90599	5	0.97632	9	0.96335	9	0.07034	-0.01298
KNN	0.87371	10	0.96679	10	0.94967	10	0.09308	-0.01712
GNB	0.81585	12	0.96121	11	0.93946	11	0.14537	-0.02175
DTC	0.81598	11	0.94654	12	0.91970	12	0.13055	-0.02683
BNB	0.70345	13	0.91903	13	0.88229	13	0.21557	-0.03674

Table 9: Normalized Recall Best Fit Line and MSE

Model	m Recall	m rank	b Recall	b rank	MSE Recall	MSE rank
SVM-rbf	-0.03187	1	98.71981	2	0.23976	1
SVM-l	-0.04159	2	98.81067	1	0.24997	2
RF	-0.05680	3	97.25393	8	0.79606	7
AdaBoost	-0.05964	4	97.58481	6	0.70904	5
Ensemble	-0.06273	5	97.11153	9	0.92620	8
SGD	-0.06815	6	97.92868	5	1.83304	10
KNN	-0.07617	7	95.98267	10	1.65432	9
LSVC	-0.07829	8	98.19906	3	0.46340	3
LDA	-0.08045	9	98.01390	4	0.55118	4
LR	-0.08642	10	97.52860	7	0.76930	6
GNB	-0.10219	11	95.36164	11	2.36384	11
DTC	-0.13130	12	93.75032	12	3.84964	12
BNB	-0.18357	13	90.37296	13	7.67408	13

recall = (mx+b)/100

4.3.5 Standardized Recall

Regarding recall for the standardized approach, Figure 16 again reveals that observations from the Non-Rescaled and Normalized approaches hold true: (a) performance jumps from $train_0$ to $test_1$,; (b) rank-orderings generally remain the same in test weeks; and, (c) recall seems to mirror the same highs and lows of the malicious URLs reported each week, which can be seen in the Malicious URL Counts from section 4.3.2.

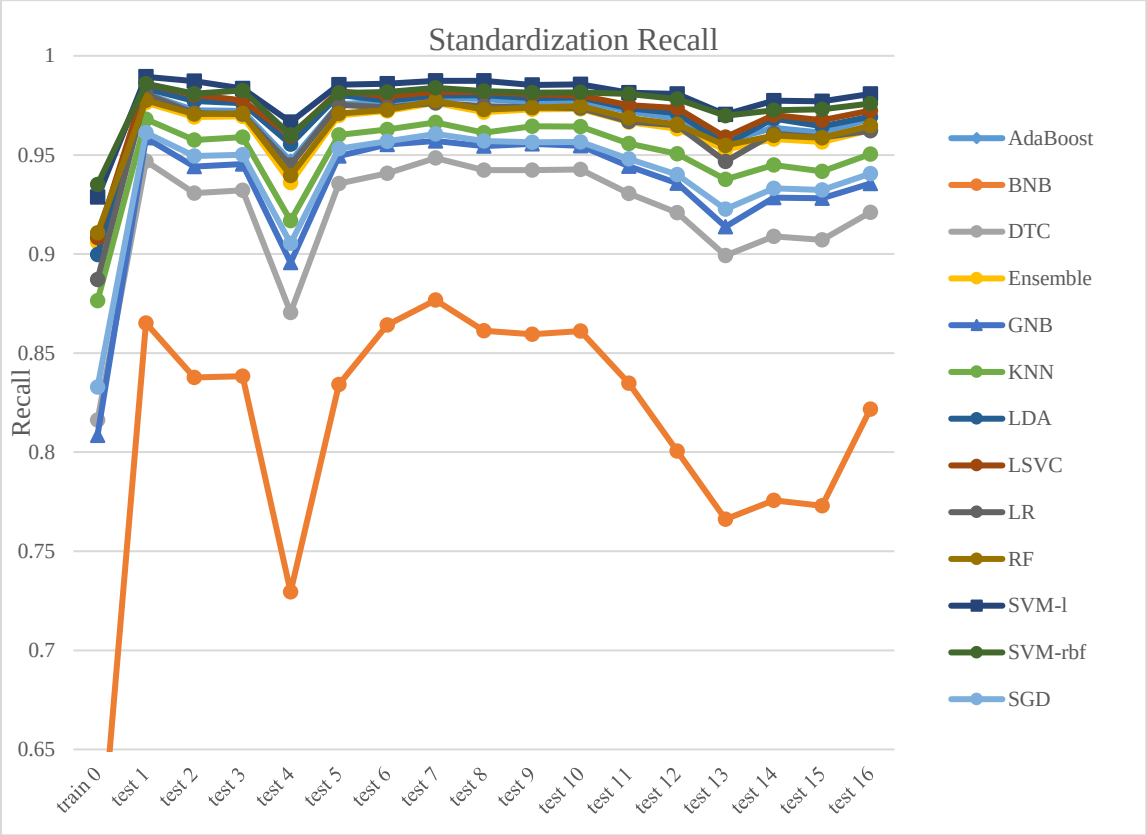


Figure 16: Standardized Recall Temporal Graph

Table 10 shows that each standardized model's recall rank-ordering among $train_0$, $test_1$, and $test_{16}$ can be reviewed. When looking to the top four performers between $train_0$ and $test_1$, trade in rank can again be observed: LSVC and LDA jump from ranks five and seven to ranks three and four, respectively. RF dropped from rank three in $train_0$ down to

rank seven in $test_1$, and AdaBoost dropped from rank four in $train_0$ down to rank five in $test_1$. On the contrary, LSVC, LDA, RF, and AdaBoost are again within a similar margin of recall (an approximate 0.8% maximum difference) in test weeks one and 16. Table 11 shows the line of best fit for each standardized model's recall.

Table 10: Standardized Recall Rank-Order

Model	train Recall	train rank	test1 Recall	test1 rank	test16 Recall	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.92869	2	0.98940	1	0.98081	1	0.06072	-0.00859
SVM-rbf	0.93512	1	0.98591	2	0.97595	2	0.05079	-0.00995
LSVC	0.90823	5	0.98580	3	0.97244	3	0.07757	-0.01336
LDA	0.89962	7	0.98350	4	0.96932	4	0.08388	-0.01417
AdaBoost	0.90948	4	0.98091	5	0.96817	5	0.07144	-0.01275
RF	0.91069	3	0.97752	7	0.96475	6	0.06682	-0.01276
Ensemble	0.90606	6	0.97636	8	0.96257	7	0.07030	-0.01378
LR	0.88716	8	0.98069	6	0.96224	8	0.09353	-0.01844
KNN	0.87639	9	0.96804	9	0.95040	9	0.09165	-0.01764
SGD	0.83285	10	0.96120	10	0.94060	10	0.12836	-0.02061
GNB	0.80852	12	0.95862	11	0.93548	11	0.15010	-0.02314
DTC	0.81617	11	0.94683	12	0.92103	12	0.13066	-0.02580
BNB	0.57702	13	0.86510	13	0.82168	13	0.28808	-0.04342

Table 11: Standardized Recall Best Fit Line and MSE

Model	m Recall	m rank	b Recall	b rank	MSE Recall	MSE rank
SVM-rbf	-0.04552	1	98.21058	3	0.36067	2
SVM-l	-0.05239	2	98.64362	1	0.32420	1
RF	-0.05851	3	97.20121	7	0.84580	7
AdaBoost	-0.06223	4	97.56963	5	0.72003	5
Ensemble	-0.06388	5	97.08392	8	0.95201	8
KNN	-0.07973	6	96.06224	9	1.54393	9
LSVC	-0.07977	7	98.22154	2	0.45961	3
LDA	-0.08108	8	97.99790	4	0.55513	4
LR	-0.09504	9	97.52812	6	0.80684	6
SGD	-0.10449	10	95.40732	10	13.00645	12
GNB	-0.10867	11	95.02232	11	2.64913	10
DTC	-0.12660	12	93.70087	12	3.81307	11
BNB	-0.33557	13	85.34702	13	15.39841	13

recall = (mx+b)/100

4.3.6 Collective Recall Rank-Order

Table 12 combines Tables 6, 8, and 10. Knowing that models do not generally change in their recall rank-ordering between each test week, as observed in the prior section in the temporal plots, this section will review best performers by their recall in $test_{16}$.

Normalization produced the best models for SVM-l, SVM-rbf, SGD, AdaBoost, RF, LR, GNB, and BNB. Standardization produced the best models for LSVC, and KNN. Applying neither led to the best models for LDA, Ensemble, and DTC. Non-rescaling producing the best Ensemble was to be expected, though, as the models selected for Ensemble were selected based on the non-rescaled performance. These best performers for each scaling-model pair are denoted in Table 12, in bold underlining.

Table 12: Collective Recall Rank-Order

Type	Model	train Recall	train rank	test1 Recall	test1 rank	test16 Recall	test16 rank	train to test1 delta	test1 to test16 delta
Normalized	**SVM-l**	**0.93510**	**3**	**0.99059**	**1**	**0.98351**	**1**	**0.05550**	**-0.00708**
Normalized	**SVM-rbf**	**0.94224**	**1**	**0.99024**	**2**	**0.98312**	**2**	**0.04800**	**-0.00712**
Non-Rescaled	SVM-l	0.93204	4	0.98988	3	0.98183	3	0.05784	-0.00805
Standardized	SVM-l	0.92869	5	0.98940	4	0.98081	4	0.06072	-0.00859
Non-Rescaled	SVM-rbf	0.92458	6	0.98724	5	0.97676	5	0.06266	-0.01048
Standardized	SVM-rbf	0.93512	2	0.98591	6	0.97595	6	0.05079	-0.00995
Standardized	**LSVC**	**0.90823**	**13**	**0.98580**	**7**	**0.97244**	**7**	**0.07757**	**-0.01336**
Normalized	LSVC	0.90510	17	0.98534	8	0.97190	8	0.08025	-0.01344
Normalized	**SGD**	**0.90242**	**18**	**0.98371**	**9**	**0.97125**	**9**	**0.08129**	**-0.01246**
Non-Rescaled	**LDA**	**0.89937**	**20**	**0.98367**	**10**	**0.96948**	**10**	**0.08430**	**-0.01420**
Standardized	LDA	0.89962	19	0.98350	12	0.96932	11	0.08388	-0.01417
Normalized	LDA	0.89897	21	0.98367	11	0.96932	12	0.08470	-0.01435
Normalized	**AdaBoost**	**0.90828**	**12**	**0.98159**	**13**	**0.96857**	**13**	**0.07331**	**-0.01302**
Non-Rescaled	AdaBoost	0.90783	14	0.98111	14	0.96852	14	0.07328	-0.01259
Standardized	AdaBoost	0.90948	11	0.98091	15	0.96817	15	0.07144	-0.01275
Normalized	**RF**	**0.91216**	**7**	**0.97770**	**21**	**0.96552**	**16**	**0.06553**	**-0.01218**
Non-Rescaled	RF	0.91103	8	0.97787	19	0.96544	17	0.06684	-0.01243
Non-Rescaled	**Ensemble**	**0.90949**	**10**	**0.97784**	**20**	**0.96513**	**18**	**0.06835**	**-0.01271**
Standardized	RF	0.91069	9	0.97752	22	0.96475	19	0.06682	-0.01276
Normalized	**LR**	**0.88286**	**23**	**0.97970**	**17**	**0.96366**	**20**	**0.09684**	**-0.01604**
Normalized	Ensemble	0.90599	16	0.97632	24	0.96335	21	0.07034	-0.01298
Non-Rescaled	LR	0.87726	24	0.97825	18	0.96282	22	0.10099	-0.01543
Standardized	Ensemble	0.90606	15	0.97636	23	0.96257	23	0.07030	-0.01378
Standardized	LR	0.88716	22	0.98069	16	0.96224	24	0.09353	-0.01844
Standardized	**KNN**	**0.87639**	**25**	**0.96804**	**25**	**0.95040**	**25**	**0.09165**	**-0.01764**
Normalized	KNN	0.87371	26	0.96679	26	0.94967	26	0.09308	-0.01712
Non-Rescaled	LSVC	0.86859	27	0.95922	30	0.94235	27	0.09063	-0.01687
Standardized	SGD	0.83285	29	0.96120	29	0.94060	28	0.12836	-0.02061
Normalized	**GNB**	**0.81585**	**32**	**0.96121**	**28**	**0.93946**	**29**	**0.14537**	**-0.02175**
Standardized	GNB	0.80852	34	0.95862	31	0.93548	30	0.15010	-0.02314
Non-Rescaled	GNB	0.79435	35	0.95548	32	0.92982	31	0.16113	-0.02566
Non-Rescaled	KNN	0.86002	28	0.96437	27	0.92662	32	0.10435	-0.03775
Non-Rescaled	**DTC**	**0.81427**	**33**	**0.94725**	**33**	**0.92146**	**33**	**0.13298**	**-0.02579**
Standardized	DTC	0.81617	30	0.94683	34	0.92103	34	0.13066	-0.02580
Normalized	DTC	0.81598	31	0.94654	35	0.91970	35	0.13055	-0.02683
Normalized	**BNB**	**0.70345**	**36**	**0.91903**	**36**	**0.88229**	**36**	**0.21557**	**-0.03674**
Non-Rescaled	BNB	0.69059	37	0.91624	37	0.87853	37	0.22565	-0.03771
Standardized	BNB	0.57702	39	0.86510	38	0.82168	38	0.28808	-0.04342
Non-Rescaled	SGD	0.60286	38	0.83778	39	0.79117	39	0.23493	-0.04661

Of the top performers in $test_{16}$, all three SVM-l, all three SVM-rbf, and LSVC (Standardized and Normalized) comprise the top eight. Although LSVC was the seventh fastest model to train and test, SVM-rbf and SVM-l took the longest, so the time taken for their models to learn was not in vain. While SVM-l and SVM-r were within the top six even during training, LSVC for standardization and normalization were of ranks 13 and 17, respectively.

Ranks nine through 13 for $test_{16}$ reveal additional trades in rank. All three LDA approaches jumped from ranks 19, 20, and 21 to ranks 10, 11, and 12. SGD with normalization jumped from rank 18 in training up to rank nine. Although SGD with normalization made a leap in rank, SGD's counterparts for standardization and non-rescaling can be found in the bottom 12 performers. The bottom 12 performers also include all methods for each GNB, DTC, BNB, and the Non-Rescaled KNN.

4.3.7 Collective Recall Best Fit Line and MSE

Table 13 combines Tables 7, 9, and 11. In terms of slope (m), normalization produced the best models for each learner except for Ensemble and DTC, which had the best slopes when rescaling was not applied. Likewise, SVM-rbf and SVM-l can be found in the top six ranks in table 12, which corresponds with some of the best y-intercepts (b) and MSE in table 13. The models with the highest slope for each scaling-model pair are denoted in Table 13, in bold underlining.

The reason for sorting by the slope rather than something else is twofold. First, Table 12 already reviewed the best performance in terms of raw recall. Second, if there is a concern that lower volumes of malicious URLs result in decreased recall, then models with slope nearer to zero should be less affected, provided that the MSE is acceptable.

Table 13: Collective Recall Best Fit Line and MSE

Type	Model	m Recall	m rank	b Recall	b rank	MSE Recall	MSE rank
Normalized	**SVM-rbf**	**-0.03187**	**1**	**98.71981**	**2**	**0.23976**	**1**
Normalized	**SVM-l**	**-0.04159**	**2**	**98.81067**	**1**	**0.24997**	**2**
Non-Rescaled	SVM-rbf	-0.04191	3	98.28780	5	0.78574	16
Standardized	SVM-rbf	-0.04552	4	98.21058	7	0.36067	5
Non-Rescaled	SVM-l	-0.04697	5	98.71208	3	0.28942	3
Standardized	SVM-l	-0.05239	6	98.64362	4	0.32420	4
Normalized	**RF**	**-0.05680**	**7**	**97.25393**	**21**	**0.79606**	**18**
Non-Rescaled	**Ensemble**	**-0.05687**	**8**	**97.27078**	**19**	**0.79111**	**17**
Non-Rescaled	RF	-0.05831	9	97.26260	20	0.76671	14
Standardized	RF	-0.05851	10	97.20121	22	0.84580	20
Normalized	**AdaBoost**	**-0.05964**	**11**	**97.58481**	**13**	**0.70904**	**11**
Non-Rescaled	AdaBoost	-0.06138	12	97.56030	15	0.73845	13
Standardized	AdaBoost	-0.06223	13	97.56963	14	0.72003	12
Normalized	Ensemble	-0.06273	14	97.11153	23	0.92620	21
Standardized	Ensemble	-0.06388	15	97.08392	24	0.95201	22
Normalized	**SGD**	**-0.06815**	**16**	**97.92868**	**12**	**1.83304**	**27**
Normalized	**KNN**	**-0.07617**	**17**	**95.98267**	**27**	**1.65432**	**26**
Normalized	**LSVC**	**-0.07829**	**18**	**98.19906**	**8**	**0.46340**	**7**
Standardized	KNN	-0.07973	19	96.06224	26	1.54393	25
Standardized	LSVC	-0.07977	20	98.22154	6	0.45961	6
Normalized	**LDA**	**-0.08045**	**21**	**98.01390**	**10**	**0.55118**	**8**
Non-Rescaled	LDA	-0.08088	22	98.01952	9	0.55123	9
Standardized	LDA	-0.08108	23	97.99790	11	0.55513	10
Normalized	**LR**	**-0.08642**	**24**	**97.52860**	**16**	**0.76930**	**15**
Non-Rescaled	LSVC	-0.08733	25	95.35129	30	1.28780	24
Non-Rescaled	LR	-0.08858	26	97.30106	18	1.00635	23
Standardized	LR	-0.09504	27	97.52812	17	0.80684	19
Normalized	**GNB**	**-0.10219**	**28**	**95.36164**	**29**	**2.36384**	**29**
Standardized	SGD	-0.10449	29	95.40732	28	13.00645	38
Standardized	GNB	-0.10867	30	95.02232	31	2.64913	30
Non-Rescaled	**DTC**	**-0.11745**	**31**	**93.68382**	**35**	**3.75421**	**32**
Non-Rescaled	GNB	-0.12347	32	94.61692	32	3.05450	31
Standardized	DTC	-0.12660	33	93.70087	34	3.81307	33
Normalized	DTC	-0.13130	34	93.75032	33	3.84964	34
Non-Rescaled	KNN	-0.17849	35	96.13513	25	2.23285	28
Normalized	**BNB**	**-0.18357**	**36**	**90.37296**	**36**	**7.67408**	**35**
Non-Rescaled	BNB	-0.18512	37	90.04325	37	7.99710	36
Non-Rescaled	SGD	-0.21914	38	81.54315	39	11.76663	37
Standardized	BNB	-0.33557	39	85.34702	38	15.39841	39

4.3.8 Correlation Analysis

Addressing why recall seems to correlate to the T_i of malicious URLs reported each week, a Spearman's rank correlation test was found to be an appropriate method for demonstrating the relation as outlined in Table 14. A Spearman's correlation matrix among all T_i and all $recall_i$ was foregone, as scatterplots demonstrated that few features held a monotonic correlation. However, scatterplots showing the monotonic relationship

between the T_i for the count of malicious URLs reported each week, and each model's $recall_i$ can be found in Appendix B.

Table 14: Spearman's Rank-Order Correlation Coefficients and p-values

Type	Model	R_s	T-Test	p-value
Normalized	SVM-l	0.882	7.002962	0.000006218736015
Standardized	LSVC	0.891	7.343155	0.000003658153580
Non-Rescaled	SVM-l	0.897	7.592839	0.000002501959599
Standardized	SVM-l	0.9	7.725556	0.000002051089863
Normalized	LSVC	0.909	8.160265	0.000001086282509
Standardized	SVM-rbf	0.909	8.160265	0.000001086282509
Standardized	LDA	0.921	8.84599	0.000000417066718
Non-Rescaled	LDA	0.921	8.84599	0.000000417066718
Non-Rescaled	SVM-rbf	0.924	9.041212	0.000000320657065
Normalized	LDA	0.929	9.39257	0.000000201849363
Normalized	SVM-rbf	0.929	9.39257	0.000000201849363
Standardized	LR	0.935	9.864577	0.000000110571817
Normalized	LR	0.953	11.76947	0.000000011993018
Standardized	BNB	0.953	11.76947	0.000000011993018
Non-Rescaled	LR	0.953	11.76947	0.000000011993018
Normalized	GNB	0.959	12.66116	0.000000004684829
Non-Rescaled	KNN	0.962	13.1825	0.000000002774388
Standardized	GNB	0.968	14.43284	0.000000000846583
Standardized	KNN	0.971	15.19641	0.000000000428419
Non-Rescaled	GNB	0.971	15.19641	0.000000000428419
Normalized	RF	0.974	16.08654	0.000000000201073
Normalized	SGD	0.974	16.08654	0.000000000201073
Standardized	SGD	0.974	16.08654	0.000000000201073
Normalized	KNN	0.976	16.76929	0.000000000115431
Standardized	AdaBoost	0.976	16.76929	0.000000000115431
Standardized	Ensemble	0.976	16.76929	0.000000000115431
Non-Rescaled	RF	0.976	16.76929	0.000000000115431
Normalized	BNB	0.979	17.9686	0.000000000045691
Non-Rescaled	AdaBoost	0.979	17.9686	0.000000000045691
Non-Rescaled	BNB	0.979	17.9686	0.000000000045691
Non-Rescaled	Ensemble	0.979	17.9686	0.000000000045691
Standardized	RF	0.982	19.45304	0.000000000015655
Normalized	DTC	0.985	21.35868	0.000000000004404
Normalized	Ensemble	0.985	21.35868	0.000000000004404
Non-Rescaled	LSVC	0.985	21.35868	0.000000000004404
Normalized	AdaBoost	0.991	27.70006	0.000000000000125
Standardized	DTC	0.991	27.70006	0.000000000000125
Non-Rescaled	DTC	0.994	34.00261	0.000000000000007
Non-Rescaled	SGD	1	infinity	0

An immediate observation is all p-values are clearly under an α of .05. In fact, the largest p-value was 0.000006218736015. With all R_s being positive, and with all p-values being under .05, the conclusion is that all models hold a strong positive monotonic

correlation to the count of malicious URLs reported each week. In the case of Non-Rescaled SGD, the monotonic association is perfectly positive. An additional observation is that the top performers (SVM-l and SVM-rbf) from sections 4.3.6 and 4.3.7 generally have the highest p-values, although they are still under the decision boundary of .05.

However, a question remains as to why this correlation is the case. One possibility is that there are some malicious URLs that are easy to classify and other malicious URLs that are difficult to classify, and in weeks when there are fewer malicious URLs reported, there are fewer easy to classify malicious URLs. If this idea is true, then the models may be reflecting a degree of overfitting against easy-to-classify URLs during training. Further research would be complicated to execute, though. Trying to print out feature weights for models is not supported by Scikit-Learn and would not be logical, given that many models are not based strictly on feature weights.

4.3.9 Welch's ANOVA for Rescaling Techniques in Test Week 16

With each scaling-model pair trained 20 times and tested across the test weeks, and with 13 scaling-model pairs, the categories of this Welch's ANOVA used the rescaling technique of normalization, standardization, and non-rescaled. Grouping in this way sums to 260 responses for each dependent variable of the training validation slice and for each test week. Of those dependent variables, the recall for $test_{16}$ was selected for analysis as a matter of principle since the recall of $test_{16}$ was the furthest away in time from the initial training bucket.

Regarding assumptions for this Welch's ANOVA, outliers were removed for each rescaling category based on interquartile range outlier fences, with normal distributions confirmed for each category using Jarque-Bera tests. Sample categories were of roughly

equal size. Concerning homogeneity, a Levene's Test demonstrated that the three categories do not share equal variances, shown in Figure 17, as the p-value of Levene's test are less than an α of .05.

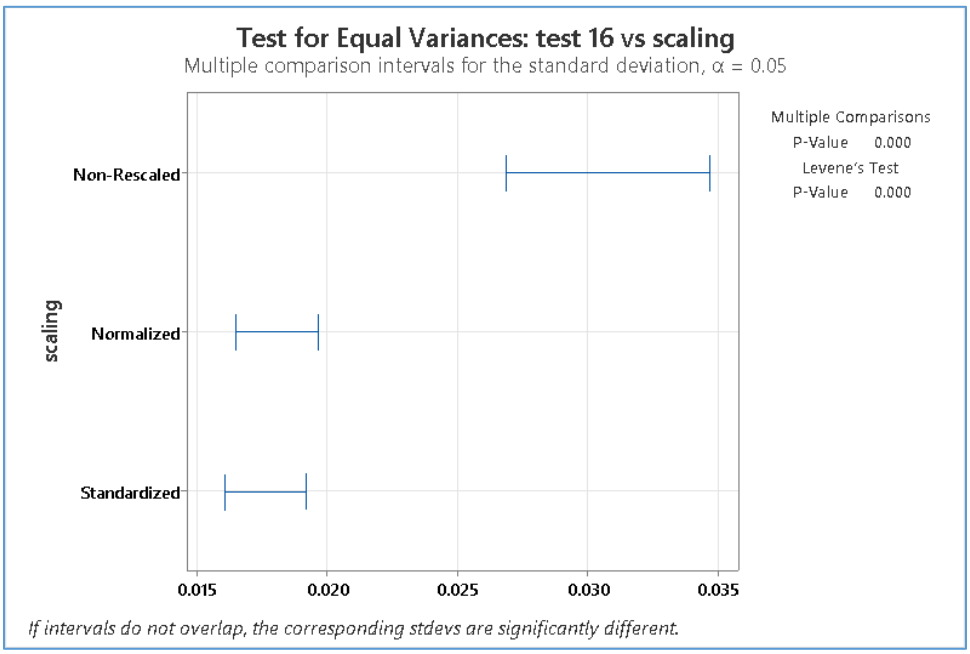
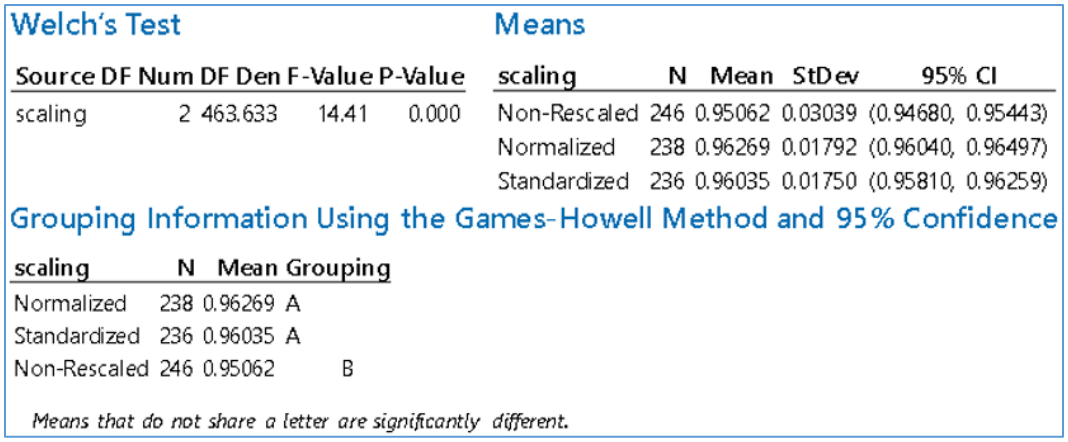


Figure 17: Lavene's Test Showing Lack of Homogeneity for Rescaling Recalls in $Test_{16}$

With assumptions for Welch's ANOVA met, Figure 18 shares the findings that: (a) there is significant difference between the mean recall of rescaling techniques in $test_{16}$, as the p-value for Welch's test is less than the α of .05; (b) normalization and standardization have insignificant difference between themselves on their affect to recall in $test_{16}$, demonstrated by the Game-Howell grouping method; and (c) normalization and standardization have a significant different and better recalls compared to the recall of non-rescaling in $test_{16}$, also demonstrated by the Game-Howell grouping method.



Welch's Test

Source	DF Num	DF Den	F-Value	P-Value
scaling	2	463.633	14.41	0.000

Means

scaling	N	Mean	StDev	95% CI
Non-Rescaled	246	0.95062	0.03039	(0.94680, 0.95443)
Normalized	238	0.96269	0.01792	(0.96040, 0.96497)
Standardized	236	0.96035	0.01750	(0.95810, 0.96259)

Grouping Information Using the Games-Howell Method and 95% Confidence

scaling	N	Mean	Grouping
Normalized	238	0.96269	A
Standardized	236	0.96035	A
Non-Rescaled	246	0.95062	B

Means that do not share a letter are significantly different.

Figure 18: Results of Welch's ANOVA for Rescaling Recalls in $Test_{16}$

4.3.10 Welch's ANOVA for Scaling-Model Pairs in Test Week 16

This section reviews a Welch's ANOVA categorized by scaling-model pairs. As each model was trained and tested over 20 iterations for each rescaling technique, each scaling-model pair category has 20 responses for each dependent variable. The recall for $test_{16}$ was again selected for analysis as a matter of principle, since the recall of $test_{16}$ was the furthest away in time from the initial training bucket.

Regarding assumptions for this Welch's ANOVA, outliers were removed for each rescaling category based on interquartile range outlier fences ranges, with normal distributions confirmed for each category using Jarque-Bera tests. Sample categories were of roughly equal size. Concerning homogeneity, a Levene's Test demonstrated that the 39 scaling-model pairs did not share equal variances, shown in Figure 19, as the p-value of Levene's test are less than an α of .05. As such, Welch's ANOVA was selected.

With assumptions for Welch's ANOVA met, Welch's ANOVA was performed. While there was significant difference between scaling-model pairs, the Games-Howell Method in Minitab attempted to place three scaling-model pairs into every single group: Non-Rescaled LSVC, Normalized SGD, and Standardized SGD. To avoid confusion by

their inclusion, the Welch's ANOVA of this section was performed again and excluded those three pairs. A repeat of Levene's Test again showed that remaining 36 scaling-model pairs did not share equal variances; Welch's ANOVA was then performed again.

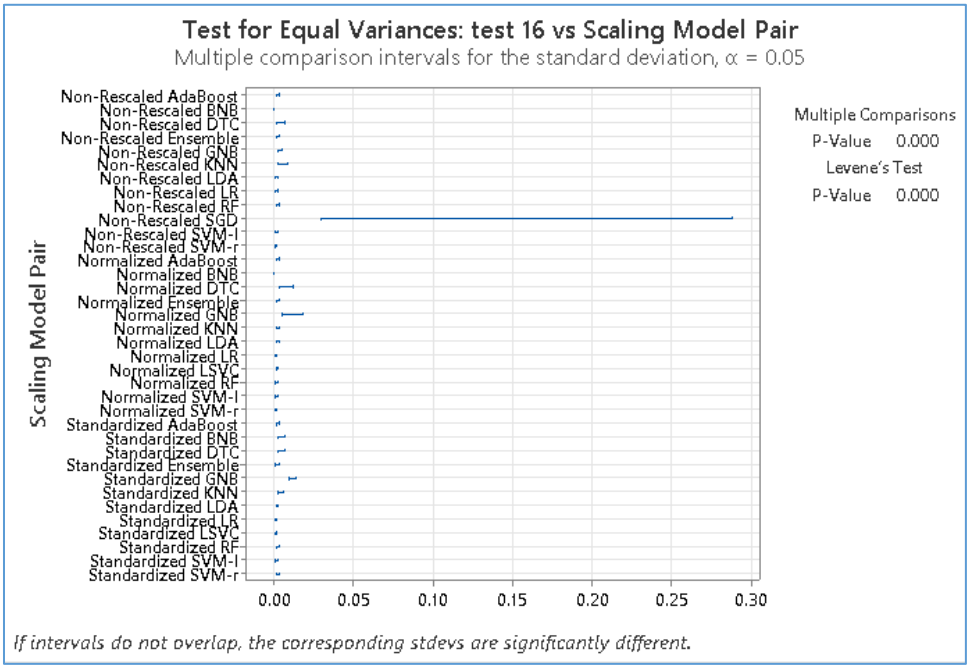


Figure 19: Lavene's Test Showing Lack of Homogeneity for Scaling-Model Pair Recalls in $Test_{16}$

Figure 20 shows that there is significant difference between the mean recall of scaling-model pairs in $test_{16}$, as the p-value for Welch's test is less than the α of .05. So to remain consistent with reported findings of Welch's ANOVA in section 4.3.9, Figure 21 shows the Means output. Figure 22 shares the Games-Howell groupings for each scaling-model pair, with line perforations added to help distinguish group separation.

Welch's Test

Source	DF Num	DF Den	F-Value	P-Value
Scaling Model Pair	35	229.210	51057.85	0.000

Figure 20: Welch's Test for Scaling-Model Pair Recalls in $Test_{16}$

Means

Scaling Model Pair	N	Mean	StDev	95% CI
Non-Rescaled AdaBoost	20	0.968518	0.002123	(0.967524, 0.969511)
Non-Rescaled BNB	18	0.885568	0.000044	(0.885546, 0.885590)
Non-Rescaled DTC	19	0.921999	0.003208	(0.920453, 0.923545)
Non-Rescaled Ensemble	20	0.965131	0.001897	(0.964243, 0.966019)
Non-Rescaled GNB	19	0.929266	0.003107	(0.927768, 0.930763)
Non-Rescaled KNN	20	0.926623	0.004280	(0.924620, 0.928626)
Non-Rescaled LDA	20	0.969475	0.001517	(0.968765, 0.970185)
Non-Rescaled LR	20	0.962822	0.001299	(0.962214, 0.963430)
Non-Rescaled RF	20	0.965436	0.002074	(0.964466, 0.966407)
Non-Rescaled SGD	17	0.8548	0.0807	(0.8133, 0.8963)
Non-Rescaled SVM-l	20	0.981826	0.001369	(0.981186, 0.982467)
Non-Rescaled SVM-r	20	0.976759	0.000915	(0.976331, 0.977188)
Normalized AdaBoost	20	0.968570	0.001850	(0.967705, 0.969436)
Normalized BNB	18	0.885584	0.000051	(0.885558, 0.885610)
Normalized DTC	19	0.92066	0.00582	(0.91785, 0.92346)
Normalized Ensemble	20	0.963348	0.001765	(0.962522, 0.964174)
Normalized GNB	20	0.93946	0.00880	(0.93534, 0.94358)
Normalized KNN	18	0.950176	0.001700	(0.949331, 0.951021)
Normalized LDA	20	0.969316	0.001865	(0.968444, 0.970189)
Normalized LR	18	0.963963	0.000951	(0.963491, 0.964436)
Normalized LSVC	20	0.971901	0.001435	(0.971230, 0.972573)
Normalized RF	19	0.965249	0.001423	(0.964563, 0.965934)
Normalized SVM-l	17	0.983805	0.001191	(0.983192, 0.984417)
Normalized SVM-r	19	0.983260	0.000699	(0.982924, 0.983597)
Standardized AdaBoost	20	0.968166	0.002018	(0.967222, 0.969111)
Standardized BNB	20	0.821683	0.003578	(0.820008, 0.823357)
Standardized DTC	19	0.921840	0.003298	(0.920250, 0.923430)
Standardized Ensemble	19	0.962959	0.001591	(0.962192, 0.963725)
Standardized GNB	20	0.93548	0.01007	(0.93077, 0.94019)
Standardized KNN	20	0.950403	0.003354	(0.948834, 0.951973)
Standardized LDA	20	0.969323	0.001591	(0.968578, 0.970068)
Standardized LR	20	0.962242	0.001001	(0.961774, 0.962710)
Standardized LSVC	19	0.972677	0.001007	(0.972192, 0.973162)
Standardized RF	18	0.964871	0.001936	(0.963908, 0.965834)
Standardized SVM-l	20	0.980809	0.001426	(0.980141, 0.981476)
Standardized SVM-r	19	0.975696	0.001835	(0.974811, 0.976580)

Figure 21: Means Results of Welch's ANOVA for Scaling-Model Pair Recalls in $Test_{16}$

Grouping Information Using the Games-Howell Method and 95% Confidence

Scaling Model Pair	N	Mean	Grouping
Normalized SVM-l	17	0.983805	A
Normalized SVM-r	19	0.983260	A B
Non-Rescaled SVM-l	20	0.981826	B C
Standardized SVM-l	20	0.980809	C
Non-Rescaled SVM-r	20	0.976759	D
Standardized SVM-r	19	0.975696	D
Standardized LSVC	19	0.972677	E
Normalized LSVC	20	0.971901	E
Non-Rescaled LDA	20	0.969475	F
Standardized LDA	20	0.969323	F
Normalized LDA	20	0.969316	F
Normalized AdaBoost	20	0.968570	F
Non-Rescaled AdaBoost	20	0.968518	F
Standardized AdaBoost	20	0.968166	F
Non-Rescaled RF	20	0.965436	G
Normalized RF	19	0.965249	G
Non-Rescaled Ensemble	20	0.965131	G H
Standardized RF	18	0.964871	G H I
Normalized LR	18	0.963963	G H I
Normalized Ensemble	20	0.963348	G H I J
Standardized Ensemble	19	0.962959	H I J
Non-Rescaled LR	20	0.962822	I J
Standardized LR	20	0.962242	J
Standardized KNN	20	0.950403	K
Normalized KNN	18	0.950176	K
Normalized GNB	20	0.93946	L
Standardized GNB	20	0.93548	L M
Non-Rescaled GNB	19	0.929266	M
Non-Rescaled KNN	20	0.926623	M N
Non-Rescaled DTC	19	0.921999	N
Standardized DTC	19	0.921840	N
Normalized DTC	19	0.92066	N
Normalized BNB	18	0.885584	O
Non-Rescaled BNB	18	0.885568	O
Non-Rescaled SGD	17	0.8548	L M N O P
Standardized BNB	20	0.821683	P

Means that do not share a letter are significantly different.

Figure 22: Games-Howell Grouping of Welch's ANOVA for Scaling-Model Pair Recall in $Test_{16}$

Evident by Figures 20 through 22 are several findings worth observing. In spite of scaling-model pairs appearing to have very small differences between their visual recall

in $test_{16}$ within the temporal plots (Figures 14, 15, and 16), Figure 20 proves there is statistically significant difference between them. Figure 22 proves the recall of Normalized SVM-l in $test_{16}$ is significantly different and better than all other scaling-model pairs except for Normalized SVM-r in $test_{16}$, and that the recall of SVM-l and SVM-rbf monopolize the top 4 Games-Howell groupings. Looking back to Table 5, SVM-l and SVM-rbf are the slowest learners. If timeliness of learning is a concern, Standardized LSVC and Standardized SVC would be the next best performers, and insignificantly different from each other according to Figure 22.

4.4 Addressing Research Questions and Hypotheses

The research questions and hypotheses from Section 1.6 were developed in sets, with four sets in total. This section will reiterate each, review findings relevant to address them, and declare their acceptance or rejection.

4.4.1 Research Question and Hypothesis, Set 1

RQ_1: Over time, do ML models trained to classify malicious URLs exhibit decay or drift in performance?

$H_{1\text{-}n}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that all models exhibit no drift and no decay in performance over time.

$H_{1\text{-}a}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that there are models with drift or decay in performance over time.

Recall does fluctuate across each week, as does feature proportionality among malicious URLs each week, evident in Figures 10 through 16. Those fluctuations are sufficient to state that recall does drift. As such, the null hypothesis of this set ($H_{1\text{-}n}$) is

rejected, and the alternate hypothesis of this set ($H_{1\text{-}a}$) is accepted. It can be said that of ML models created to lexically classify malicious URLs, a temporal analysis shows that there are models with drift or decay in performance over time. This additionally serves to validate the idea that a temporal analysis is a warranted technique for analyzing malicious URLs.

4.4.2 Research Question and Hypothesis, Set 2

RQ_2: For malicious URL ML models that demonstrate drift or decay, are there models with stronger lasting power?

$H_{2\text{-}n}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that all models exhibit equal lasting power.

$H_{2\text{-}a}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that there are models that exhibit different lasting power.

The temporal plots across Figures 14, 15, and 16 all confirm different degrees of lasting power. Looking to the lines of best fit, models with perfect lasting power would have an MSE and slope of zero, or nearly zero. Evident is that no model's line of best fit has the same MSE, nor a slope of zero. Weeks 4 and 13 in Figures 14, 15, and 16 present a challenge that prevents the acceptance of $H_{2\text{-}n}$. As such, the null hypothesis of this set ($H_{2\text{-}n}$) is rejected, and the alternate hypothesis of this set ($H_{2\text{-}a}$) is accepted. It can be said that Of ML models created to lexically classify malicious URLs, a temporal analysis shows that there are models that exhibit different lasting power.

Considering RQ_2, although recall correlates to the count of malicious URLs reported each week, some models correlate more strongly than others. SVM-l and SVM-rbf, regardless of scaling, demonstrate some of the strongest lasting power and recall

across time, particularly evident from Sections 4.3.6 and 4.3.7. Consequently, with some of the highest p-values in Section 4.3.8, their recalls are among the least correlated to the count of malicious URLs reported each week.

4.4.3 Research Question and Hypothesis, Set 3

RQ_3: Given enough time for a model's performance to drift or decay, do we observe the rank-ordering of performance change between models?

$H_{3\text{-}n}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that all models do not change in their performance rank-ordering.

$H_{3\text{-}a}$: Of ML models created to lexically classify malicious URLs, a temporal analysis shows that models do change in their performance rank-ordering.

Sometimes models do change in their rank-order. For example, in Figure 14, KNN trades in rank with LSVC and AdaBoost. In Figures 15 and 16, several models compete for ranks three through eight, with LR showing the most apparent trade in $test_{13}$. Additionally, Tables 6, 8, and 9 show several models trade in ranks from $train_0$ to $test_1$, as reviewed in Sections 4.3.3 through 4.3.5. As such, the null hypothesis of this set ($H_{3\text{-}n}$) is rejected, and the alternate hypothesis of this set ($H_{3\text{-}a}$) is accepted. It can be said that of ML models created to lexically classify malicious URLs, a temporal analysis shows that models do change in their performance rank-ordering.

Nevertheless, the primary motive for this set of RQ_3 was to determine if the temporal analysis would reveal a top performer, different from what a 70-30 split approach would reveal. To observe that SVM-l and SVM-rbf are consistently in ranks 1 and 2, in $train_0$ and $test_{1\text{-}16}$, assuming recall in $train_0$ is reflective of 70-30 split's recall, these findings confirm SVM-l and SVM-rbf remain the best performers. This suggests

that a data scientist would have made a temporally sensible decision in selecting SVM-l or SVM-rbf for production, even if their decision was based on only a 70-30 split. Welch's ANOVA and Game-Howell post hoc of section 4.3.10 also identified Normalized SVM-l and Normalized SVM-rbf to be of the highest group in $test_{16}$. If SVM methods were not used for sake of time, as they are slow learners, the temporally sensible decision is different from what a 70-30 split approach would show. Normalized RF dropped from rank 7 in the validation slice down to rank 16 in $test_{16}$, with the Welch's ANOVA and Game-Howell post hoc of section 4.3.10 showing that Normalized RF is inferior in $test_{16}$ compared to AdaBoost, LDA, and LSVC methods.

4.4.4 Research Question and Hypothesis, Set 4

RQ_4: Can rescaling data improve or reduce drift or decay in performance?

$H_{4\text{-}n}$: Of ML models created to lexically classify malicious URLs, there is no temporal performance difference among different data rescaling techniques.

$H_{4\text{-}a}$: Of ML models created to lexically classify malicious URLs, there is temporal performance difference among different data rescaling techniques.

If $H_{4\text{-}n}$ were true, the scaling-model pairs in Tables 12 and 13 would be identical. This is to say that, for any given model with normalization, standardization, and no rescaling, they would have the same recall in $train_0$, and $test_{1\text{-}16}$. Additionally, and most importantly, the Welch's ANOVA and Game-Howell post hoc of section 4.3.9 confirm that (a) normalization and standardization have insignificant difference between themselves on their affect to recall in $test_{16}$, demonstrated by the grouping method; and (b) normalization and standardization have a significant different and better recalls compared to the recall of non-rescaling in $test_{16}$. As such, the null hypothesis of this set

($H_{4\text{-}n}$) is rejected, and the alternate hypothesis of this set ($H_{4\text{-}a}$) is accepted. It can be said that of ML models created to lexically classify malicious URLs, there is temporal performance difference among different data rescaling techniques. Rescaling techniques (normalization and standardization) do have a positive effect on temporal performance. Not performing rescaling would be a mistake, at least for models based on the same lexical features selected in this praxis.

Chapter 5 - Discussion and Conclusions

5.1 Discussion

Drawing from results, this section reviews conclusions, declares contributions to the body of knowledge, and enumerates recommendations for future research.

5.2 Conclusions

In this study, a temporal analysis has, for the first time, been used to evaluate classification of malicious URLs with Machine Learning. The Welch's ANOVA and Game-Howell post hoc of section 4.3.9 confirm that in $test_{16}$, normalization and standardization produced statistically significant improvement to recall in comparison to the application of neither. The Welch's ANOVA and Game-Howell post hoc of section 4.3.10 additionally confirmed normalized SVM-l (.9827 ± .0035) and normalized SVM-rbf (.9824 ± .0019) to be the top performers. Although the SVM-l and SVM-rbf learners can additionally be observed as top performers in each temporal plot and comparison table, SVM-l and SVM-rbf were by far the slowest learners. If timeliness of an ML solution for classifying URLs is urgent, the strongest alternates to SVM-l and SVM-rbf methods can be found in in normalized LSVC (.9716 ± .0025), and standardized LSVC (.9710 ± .0031).While these recalls are certainly acceptable for classifying malicious URLs, and while Appendix A shares the remaining metrics against non-temporally stratified URLs, performance against temporally stratified benign URLs remains an open question.

To the overall aim of providing insight to researchers, results suggest that a data scientist would have made a temporally-sensible decision in selecting SVM-l or SVM-rbf

for production, even if their decision was based on a 70-30 split, at least for the features selected in this analysis. This is to say that the top performers of SVM-l or SVM-rbf from training remained the top performers during the plurality of test weeks, and that selection of different features may yield different results. Conversely, the next best temporal alternatives of Normalized and Standardized LSVC would not have been selected for using a 70-30 split, as they were of ranks 13 and 17 compared to the training validation slice.

Having SVM-l, SVM-rbf, Normalized LSVC, and Standardized LSVC identify as the strongest performers was not expected, as only one of the prior 35 research works on malicious URL classification identified SVM-l as their best learner. LSVC and SVM-rbf were not observed as the top performer in prior works. This discrepancy is not altogether strange, though, as few of the 35 prior research works have agreed on which learner produced the best results, with each work varying in their methodologies, data, and tested learners. The approach used for determining which models to include in Ensemble also did not select any of these models, instead using RF, DT, KNN, and LR. In hindsight, it is suspected that an Ensemble would have best been created with rescaling, and the use of LSVC, LDA, AdaBoost, and RF.

On the topic of URLs being non-stationary, it remains unclear whether performance decays, at least for features selected in this lexical temporal analysis. In light of the monotonic correlation of recall to the count of malicious URLs reported each week, and with fewer malicious URLs reported in the latter test weeks, decay, if any, may be negligible. However, recall and features do fluctuate in their proportionality, meaning they drift. This serves to confirm that at least malicious URLs reported to

URLHaus are non-stationary, and that the temporal analysis approach is warranted for analyzing malicious URLs. Overall, the temporal analysis did yield novel results that a 70-30 split would not have shown. There are, however, opportunities for improvement in future research.

5.3 Contributions to Body of Knowledge

This praxis contributes to the body of knowledge in the following key areas:

1. This praxis demonstrates the first temporal analysis for evaluating how well ML can classify malicious URLs. This temporal analysis addresses current gaps in knowledge of what happens to recall once ML solutions are put into use over a 16-week period. The results provide novel insight for what security engineers may want to consider or know when developing ML solutions for malicious URL classification, such that stronger confidence can be held in their models.
2. The dataset created for this praxis is one of the largest malicious and non-malicious URL datasets created to date; it is available upon request. While the dataset has little use for a temporal analysis outside of recall, it could be used in a 70-30 split analysis. Alternatively, this dataset could be used as a source in creating a new and improved temporal dataset for future research.

5.4 Recommendations for Future Research

Regarding recommendations for future research, those with the most potential for improvement are addressed here.

5.4.1 An Improved Temporal Dataset

An improved temporal dataset is desirable for addressing the concern that this praxis is entirely reliant on recall, with temporal performance for the remaining metrics

(accuracy, precision, F1, and AUC) not being known. Bridging this gap requires benign URLs to be temporally stratified.

Future research attempting a temporal analysis could source temporally stratified benign URLs from known safe outbound proxy log traffic from either a university or employer. While this would require researchers to complete additional paperwork, such as an NDA with the sourcing institution, future research with enough time would not be hindered by that limitation and should yield improved results. Collecting test URLs beyond the duration of 16 weeks could also be achieved. Malicious URLs could also be collected from the sourcing institution; however, that assumes all malicious URLs are known by the institution. Looking elsewhere for malicious URLs may also provide better variety.

PhishTank and OpenPhish could serve as strong alternates to URLHaus for sourcing temporally stratified malicious URLs. PhishTank, though, has had their user registration disabled for the entire duration of this praxis, and the author's attempts to contact them were left unanswered. A researcher with access to an existing PhishTank account is the only option for the foreseeable future for PhishTank. OpenPhish offers paid access and is an option for researchers with adequate funds.

As ip addresses have increased numbers of dots ('.') and numbers, features of ip-based URLs will have a different proportionality compared to non-ip-based URLs. To that point, a future lexical analysis may find more desirable results with two different datasets, where ip-based URLs are analyzed separately from non-ip-based URLs, with separate models for each. Using ML for filtering traffic can provide overhead challenges

on a network. In practice, it may be worth reducing a malicious and non-malicious URL dataset against a blacklist, with only the remainder being candidates for ML analysis.

5.4.2 Changes in Methodology

There are several ways future research could pivot from the work herein. Different and new features could be explored, such as checks for hexadecimal encoding, evaluation checks for legitimate TLDs, and use of nltk for identifying key words. There may also be utility in merging binary or categorical features that are either uncommon, or on their own, are inferior for feature selection in contrast to the alternates. However, merging features may be more of a combinatorics problem and possibly out of scope, depending on the hypothesis of future research, and perhaps unneeded if performance is acceptable.

Tensorflow could be used for exploring NNs, as it remains unknown how NNs temporally perform in URL classification. RSDD and HEOM (discussed in Section 2.6.2) could be performed between test buckets to formally test for drift, rather than use proportionality as was done herein. Future work could also test different approaches for possible effect on performance drift or decay by contrasting the applications of SMOTE, forgetting, oversampling, and under sampling.

References

Aalla, H. V. S., Dumpala, N. R., & Eliazer, M. (2021). *Malicious URL prediction using machine learning techniques*. Retrieved October 7, 2023, from https://www-proquest-com.proxygw.wrlc.org/docview/2564188868

About us. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/about.html

Abuadbba, A., Wang, S., Almashor, M., Ahmed, M. E., Gaire, R., Camtepe, S., & Nepal, S. (2022). Towards web phishing detection limitations and mitigation. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2204.00985

AdaBoost. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/ensemble.html#adaboost

Akanbi, O., Amiri, I., & Fazeldehkordi, E. (2014). *A machine-learning approach to phishing detection and defense*. Syngress.

Alarifi, A., Alsaleh, M., & Al-Salman, A. S. (2013). Security analysis of top visited Arabic web sites. *International Conference on Advanced Communication Technology*, 173–178. Retrieved from http://icact.org/program/full_paper_counter.asp?full_path=/upload/2013/0461/20130461_finalpaper.pdf

Aljabri, M., Alhaidari, F., Mohammad, R. M. A., Mirza, S., Alhamed, D. H., Altamimi, H. S., & Chrouf, S. M. B. (2022). An assessment of lexical, network, and content-based features for detecting malicious URLs using machine learning and deep learning models. *Computational Intelligence and Neuroscience*, *2022*, 1–14. https://doi.org/10.1155/2022/3241216

Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., & González, F. A. (2017). Classifying phishing URLs using recurrent neural networks. *2017 APWG Symposium on Electronic Crime Research (eCrime)*, 1–8. https://doi.org/10.1109/ecrime.2017.7945048

Barlow, L., Bendiaby, G., Shiaeles, S., & Savage, N. (2020). A novel approach to detect phishing attacks using binary visualization and machine learning. *2020 IEEE World Congress on Services (SERVICES)*. https://doi.org/10.1109/services48979.2020.00046

Bell, S. (2019, December 5). Catch me (on time) if you can: Understanding the effectiveness of Twitter URL blacklists. *arXiv (Cornell University)*. Retrieved from https://arxiv.org/abs/1912.02520

Bento, C. (2021, December 15). Support Vector Machines explained with Python examples. *Medium*. Retrieved from https://towardsdatascience.com/support-vector-machines-explained-with-python-examples-cb65e8172c85

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform resource identifier (URI): Generic syntax*. https://doi.org/10.17487/rfc3986

Bernoulli naive bayes. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes

Berners-Lee, T., Masinter, L., & McCahill, M. (1994). *Uniform resource locators (URL)*. https://doi.org/10.17487/rfc1738

Bhattacharjee, S. D., Talukder, A., Al-Shaer, E., & Doshi, P. (2017). Prioritized active learning for malicious URL detection using weighted text-based features. *2017*

IEEE International Conference on Intelligence and Security Informatics (ISI). https://doi.org/10.1109/isi.2017.8004883

Bodnar, D. (2023, October 6). *What is a honeypot? Different types + how they work.* Norton. Retrieved from https://us.norton.com/blog/iot/what-is-a-honeypot#

Bradburn, S. [Steven Bradburn]. (2020b, August 10). *Spearman correlation explained (Inc. Test assumptions)* [Video]. YouTube. https://www.youtube.com/watch?v=JwNwbu-g2m0

Brites, D., & Wei, M. (2019). PhishFry - A proactive approach to classify phishing sites using SCIKIT Learn. *2019 IEEE Globecom Workshops (GC Wkshps)*. https://doi.org/10.1109/gcwkshps45667.2019.9024428

Brownlee, J. (2020). *Train-test split for evaluating machine learning algorithms*. MachineLearningMastery.com. Retrieved from https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Brownlee, J. (2021a). *Random oversampling and undersampling for imbalanced classification*. MachineLearningMastery.com. Retrieved from m https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Brownlee, J. (2021b). *SMOTE for imbalanced classification with python*. MachineLearningMastery.com. Retrieved from https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Canali, D., Cova, M., Vigna, G., & Kruegel, C. (2011). Prophiler. *Proceedings of the 20th International Conference on World Wide Web*. https://doi.org/10.1145/1963405.1963436

Castillo, D. (2023). *Machine learning concept drift – What is it and five steps to deal with it.* Seldon. Retrieved from https://www.seldon.io/machine-learning-concept-drift

CEC Juniper Community. (n.d.). Retrieved from https://supportportal.juniper.net/s/article/How-many-Packets-per-Second-per-port-are-needed-to-achieve-Wire-Speed?language=en_US

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority ver-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chu, W., Zhu, B., Feng, X., Guan, X., & Cai, Z. (2013). Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs. *2013 IEEE International Conference on Communications (ICC)*. https://doi.org/10.1109/icc.2013.6654816

Classification: ROC curve and AUC. (2021). Google for developers. Retrieved from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Crişan, A., Florea, G., Halasz, L., Lemnaru, C., & Oprişa, C. (2020). Detecting malicious URLs based on machine learning algorithms and word embeddings. *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. https://doi.org/10.1109/iccp51029.2020.9266139

Cui, Q., Jourdan, G., Von Bochmann, G., Couturier, R., & Onut, I. (2017). Tracking phishing attacks over time. *Proceedings of the 26th International Conference on World Wide Web*. https://doi.org/10.1145/3038912.3052654

Das, A., Das, A., Datta, A., Si, S., & Barman, S. (2020). Deep approaches on malicious URL classification. *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. https://doi.org/10.1109/icccnt49239.2020.9225338

Daniel, T. [Research By Design]. (2017, April 11). *Foundations of ANOVA – assumptions and hypotheses for one-way ANOVA* [Video]. YouTube. https://www.youtube.com/watch?v=Y1pY86G74_c

Decision trees. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/tree.html

De Lucia, D. L. M., & Cotton, C. C. (2019, April 1). *Adversarial machine learning for cyber security*. Retrieved from http://jisar.org/2019-12/n1/JISARv12n1p26.html

Delua, J. (2021, March 12). *Supervised vs. unsupervised learning: What's the difference?* IBM Blog. Retrieved from https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

Dobolyi, D. G., & Abbasi, A. (2016). PhishMonger: A free and open source public archive of real-world phishing websites. *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. https://doi.org/10.1109/isi.2016.7745439

Edirimannage, S., Nabeel, M., Elvitigala, C., & Keppitiyagama, C. (2022). PhishChain: A decentralized and transparent system to blacklist phishing URLs. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2202.07882

El-Din, A. E., Hemdan, E. E., & El-Sayed, A. (2021). Malweb: An efficient malicious websites detection system using machine learning algorithms. *2021 International Conference on Electronic Engineering (ICEEM)*. https://doi.org/10.1109/iceem52022.2021.9480648

Ensembles: gradient boosting, random forests, bagging, voting, stacking. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees

Eshete, B., Villafiorita, A., & Weldemariam, K. (2013). BINSPECT: Holistic analysis and detection of malicious web pages. *Lecture Notes in Computer Science* (pp. 149–166). https://doi.org/10.1007/978-3-642-36883-7_10

Fan, Z. (2021). Detecting and classifying phishing websites by machine learning. *2021 3rd International Conference on Applied Machine Learning (ICAML)*. https://doi.org/10.1109/icaml54311.2021.00018

Fein, E. C. (2022). *Section 6.2: one-oay ANOVA assumptions, interpretation, and write up*. Pressbooks. Retrieved from https://usq.pressbooks.pub/statisticsforresearchstudents/chapter/one-way-anova-assumptions/

Frost, J. (2017). *Benefits of Welch's ANOVA compared to the classic one-way ANOVA*. Statistics by Jim. Retrieved from https://statisticsbyjim.com/anova/welchs-anova-compared-to-classic-one-way-anova/

Frost, J. (2019). *How to do one-way ANOVA in excel* Statistics by Jim. Retrieved from https://statisticsbyjim.com/hypothesis-testing/nonparametric-parametric-tests/

Frost, J. (2021). *Nonparametric tests vs. parametric tests*. Statistics by Jim. Retrieved from https://statisticsbyjim.com/hypothesis-testing/nonparametric-parametric-tests/

Frost, J. (2023a). *5 ways to find outliers in your data*. Statistics by Jim. Retrieved from https://statisticsbyjim.com/basics/outliers/

Frost, J. (2023b). *Mean squared Error (MSE).* Statistics by Jim. Retrieved from https://statisticsbyjim.com/regression/mean-squared-error-mse/

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 1–37. https://doi.org/10.1145/2523813

Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. *Proceedings of the 2007 ACM Workshop on Recurring Malcode*. https://doi.org/10.1145/1314389.1314391

Gaussian naive bayes. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes

Ghaleb, F. A., Alsaedi, M., Saeed, F., Ahmad, J., & Alasli, M. (2022). Cyber threat intelligence-based malicious URL detection model using ensemble learning. *Sensors*, *22*(9), 3373. https://doi.org/10.3390/s22093373

Glassner, A. (2021). *Deep learning: A Visual Approach*. National Geographic Books.

Hannousse, A., & Yahiouche, S. (2021). Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Engineering Applications of Artificial Intelligence*, *104*, 104347. https://doi.org/10.1016/j.engappai.2021.104347

Hassan, M., (2023). *ANOVA (analysis of variance) – forulas, types, and examples*. Researchmethods. Retrieved from https://researchmethod.net/anova/

He, H., & Ma, Y. (2013). *Imbalanced learning: Foundations, algorithms, and applications*. Wiley-IEEE Press.

Heinz, S. (2019, May 29). *Using reinforcement learning to play Super Mario Bros on NES using TensorFlow*. Medium. Retrieved from https://towardsdatascience.com/using-reinforcement-learning-to-play-super-mario-bros-on-nes-using-tensorflow-31281e35825

Huang, L., Joseph, A. D., Nelson, B. D., Rubinstein, B. I. P., & Tygar, J. D. (2011). Adversarial machine learning. *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. https://doi.org/10.1145/2046684.2046692

IBM. (n.d.). *What are Naive Bayes classifiers?* IBM. Retrieved from https://www.ibm.com/topics/naive-bayes

IBM. (2022). *Cost of a data breach report 2022. IBM.* Retrieved from https://www.ibm.com/downloads/cas/3R8N1DZJ

IBM Data and AI Team. (2023). *AI vs. machine learning vs. deep learning vs. neural networks: What's the difference?* IBM. Retrieved from https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Jiang, J., Chen, J., Choo, K. R., Liu, C., Liu, K., Yu, M., & Wang, Y. (2018). A deep learning based online malicious URL and DNS detection scheme. In *Springer eBooks* (pp. 438–448). https://doi.org/10.1007/978-3-319-78813-5_22

Kernel functions. (n.d.). Scikit-learn. Retrieved from

https://scikit-learn.org/stable/modules/svm.html#kernel-functions

Kumar, A. (2023). *ROC Curve & AUC Explained with Python Examples.* Vitalflux.com Retrieved from https://vitalflux.com/roc-curve-auc-python-false-positive-true-positive-rate/

KNeighborsClassifier. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn-neighbors-kneighborsclassifier

Kumi, S., Lim, C., & Lee, S. (2021). Malicious URL detection based on associative classification. *Entropy*, *23*(2), 182. https://doi.org/10.3390/e23020182

Li, J., & Wang, S. (2017). PhishBox: An approach for phishing validation and detection. *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. https://doi.org/10.1109/dasc-picom-datacom-cyberscitec.2017.101

LinearDiscriminantAnalysis. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html#sklearn.discriminant_analysis.LinearDiscriminantAnalysis

LinearSVC. (n.d.). Scikit-learn. Retrieved from

https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

Li, T., Kou, G., & Peng, Y. (2020). Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Information Systems*, *91*, 101494. https://doi.org/10.1016/j.is.2020.101494

Likebupt. (2021, November 4). *SMOTE - Azure machine learning*. Microsoft. Retrieved from https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/smote

Logistic regression. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Mamun, M. S. I., Rathore, M. A., Lashkari, A. H., Stakhanova, N., & Ghorbani, A. A. (2016). Detecting malicious URLs using lexical analysis. *Lecture Notes in Computer Science* (pp. 467–482). https://doi.org/10.1007/978-3-319-46298-1_30

Marchal, S., Jérôme, F., State, R., & Engel, T. (2014). PhishStorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, *11*(4), 458–471. https://doi.org/10.1109/tnsm.2014.2377295

Martins, N., Cruz, J. M., Cruz, T., & Abreu, P. H. (2020). Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. *IEEE Access*, *8*, 35403–35419. https://doi.org/10.1109/access.2020.2974752

Mašetić, Z., Subaşı, A., & Azemović, J. (2016). Malicious web sites detection using C4.5 Decision Tree. *Southeast Europe Journal of Soft Computing*, *5*(1). https://doi.org/10.21533/scjournal.v5i1.109

Mitra, D., & Raja M.V, R. (2021, August 13). *Identifying suspicious URLs*. Kaggle. Retrieved from https://www.kaggle.com/datasets/ruthvikrajamv/identifying-suspiciousurlsmllexical-analysis?select=Final+Report.pdf

Moore, T., & Clayton, R. (2008). Evaluating the wisdom of crowds in assessing phishing websites. *Lecture Notes in Computer Science* (pp. 16–30). https://doi.org/10.1007/978-3-540-85230-8_2

Muhati, E., & Rawat, D. B. (2021). Adversarial machine learning for inferring augmented cyber agility prediction. *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. https://doi.org/10.1109/infocomwkshps51825.2021.9484471

Oshingbesan, A., Ekoh, C., Okobi, C., Munezero, A., & Richard, K. (2022). Detection of comalicious websites using machine learning techniques. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2209.09630

Redirections in HTTP. (2023, October 5). MDN. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/Redirections

Redirects and Google search. (n.d.). Google Search Central. Retrieved from https://developers.google.com/search/docs/crawling-indexing/301-redirects

Roy, S. S. (2021, November 13). *Evaluating the effectiveness of phishing reports on Twitter*. *arXiv (Cornell University)*. Retrieved from https://arxiv.org/abs/2111.07201

Rupa, C., Srivastava, G., Bhattacharya, S., Maddikunta, P. K. R., & Gadekallu, T. R. (2021). Malicious URL detection using logistic regression. *2021 IEEE*

International Conference on Omni-Layer Intelligent Systems (COINS). https://doi.org/10.1109/coins51742.2021.9524269

Sabir, B., Babar, A., Gaire, R., & Abuadbba, A. (2022). Reliability and robustness analysis of machine learning based phishing URL detectors. *IEEE Transactions on Dependable and Secure Computing*, 1–18. https://doi.org/10.1109/tdsc.2022.3218043

Şahingöz, Ö. K., Büber, E., Demir, Ö., & DiRi, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications, 117*, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029

Schlegel, B. A. (2020). *Games-Howell post-hoc multiple comparisons test with python.* Aaron Schlegel's Notebook of Interesting Things. Retrieved from https://aaronschlegel.me/games-howell-post-hoc-multiple-comparisons-test-python.html

Sereno. (2023). *Comparison of Pearson vs Spearman correlation coefficients*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2021/03/comparison-of-pearson-and-spearman-correlation-coefficients/

Shah, T. (2021, May 24). *About train, validation and test sets in machine learning.* Medium. Retrieved from https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

Shahrivari, V., Darabi, M., & Izadi, M. (2020). Phishing detection using machine learning techniques. *arXiv (Cornell University)*. Retrieved from https://arxiv.org/abs/2009.11116

Shantanu, Janet, B., & Kumar, R. (2021). Malicious URL detection: A comparative study. *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. https://doi.org/10.1109/icais50930.2021.9396014

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(1), 41–50. https://doi.org/10.1109/tetci.2017.2772792

Simic, M. (2023). *Hard vs. soft voting classifiers*. Baeldung on Computer Science. Retrieved from https://www.baeldung.com/cs/hard-vs-soft-voting-classifiers

Singh, A. (2012). *On concept drift, deployability, and adversarial selection in machine learning-based malware detection*. ProQuest. Retrieved from http://proxygw.wrlc.org/login?url=https://www-proquest-com.proxygw.wrlc.org/dissertations-theses/on-concept-drift-deployability-adversarial/docview/1270815779/se-2

Singh, A. K. (2020). Malicious and benign webpages dataset. *Data in Brief*, *32*, 106304. https://doi.org/10.1016/j.dib.2020.106304

Singhal, S., Chawla, U., & Shorey, R. (2020). Machine learning & concept drift based approach for malicious website detection. *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. https://doi.org/10.1109/comsnets48256.2020.9027485

The size of the World Wide Web (The Internet). (2023). Retrieved from https://www.worldwidewebsize.com/

Spearman's rank-order correlation - A guide to when to use it, what it does and what the assumptions are. (n.d.). Laerd Statistics. Retrieved from https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php

Srinivasan, S., Ravi, V., Arunachalam, A., Alazab, M., & Soman, K. P. (2020). DURLD: Malicious URL detection using deep learning-based character level representations. In *Springer eBooks* (pp. 535–554). https://doi.org/10.1007/978-3-030-62582-5_21

Stochastic gradient descent. (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/sgd.html#classification

Swarnkar, M., Sharma, N., & Thakkar, H. K. (2022). Malicious URL detection using machine learning. In *Predictive Data Security using AI* (pp. 199–216). https://doi.org/10.1007/978-981-19-6290-5_11

Tan, G., Zhang, P., Liu, Q., Liu, X., Zhu, C., & Dou, F. (2018). Adaptive malicious URL detection: Learning in the presence of concept drifts. *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*. https://doi.org/10.1109/trustcom/bigdatase.2018.00107

Tanaka, S., Matsunaka, T., Yamada, A., & Kubota, A. (2021). Phishing site detection using similarity of website structure. *2021 IEEE Conference on Dependable and Secure Computing (DSC)*. https://doi.org/10.1109/dsc49826.2021.9346256

Tanner, G. (2020a). *K nearest neighbors*. Machine Learning Explained. Retrieved from https://ml-explained.com/blog/k-nearest-neighbors-explained

Tanner, G. (2020b). *Logistic regression*. Machine Learning Explained. Retrieved from https://ml-explained.com/blog/logistic-regression-explained

Tanner, G. (2021a). *Linear discriminant analysis (LDA)*. Machine Learning Explained. Retrieved from https://ml-explained.com/blog/linear-discriminant-analysis-explained

Tanner, G. (2021b). *Random forest*. Machine Learning Explained. Retrieved from https://ml-explained.com/blog/random-forest-explained

Tyagi, V. (2023). *URL feature extractor*. Kaggle. Retrieved from https://www.kaggle.com/code/vishalxtyagi/url-feature-extractor/output

Uniform resource identifier (URI) schemes. (n.d.). IANA. Retrieved from https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

URLhaus - API. (n.d.). URLhaus. Retrieved from https://urlhaus.abuse.ch/api/

Usama, M., Qadir, J., Al-Fuqaha, A., & Hamdi, M. (2020). The adversarial machine learning conundrum: Can the insecurity of ML become the Achilles' heel of cognitive networks? *IEEE Network, 34*(1), 196–203. https://doi.org/10.1109/mnet.001.1900197

Voting classifier (n.d.). Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier

Vyawhare, C. R., Totare, R. Y., Sonawane, P. S., & Deshmukh, P. B. (2022). Machine learning system for malicious website detection using concept drift detection.

International Journal for Research in Applied Science and Engineering Technology, *10*(5), 47–55. https://doi.org/10.22214/ijraset.2022.42048

Wayback machine. (n.d.). Web Archive. Retrieved from https://web.archive.org/web/20200119103319/http://hosts-file.net/ad_servers.txt

Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., & Woźniak, M. (2020). Accurate and fast URL phishing detector: A convolutional neural network approach. *Computer Networks*, *178*, 107275. https://doi.org/10.1016/j.comnet.2020.107275

What is a URL? - Learn web development. (2023, August 2). MDN. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL

Wood, T. (2020). Precision and recall. DeepAI. Retrieved from https://deepai.org/machine-learning-glossary-and-terms/precision-and-recall

Xiao, P. (2022). *Artificial intelligence programming with Python: From zero to hero*. Wiley.

Xuan, C. D., Nguyen, H. D., & Nikolaevich, T. V. (2020). Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science and Applications*, *11*(1). https://doi.org/10.14569/ijacsa.2020.0110119

Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., & Wang, G. (2021). BODMAS: An open dataset for learning based temporal analysis of PE malware. *2021 IEEE Security and Privacy Workshops (SPW)*. https://doi.org/10.1109/spw53761.2021.00020

Yeboah-Ofori, A., Ismail, U. M., Swidurski, T., & Opoku-Boateng, F. A. (2021). Cyber threat ontology and adversarial machine learning attacks: Analysis and prediction perturbance. *2021 International Conference on Computing, Computational Modelling and Applications (ICCMA)*. https://doi.org/10.1109/iccma53594.2021.00020

Yuan, J.-T., Liu, Y., & Yu, L. (2021). A novel approach for malicious URL detection based on the joint model. *Security and Communication Networks, 2021*, 1–12. https://doi.org/10.1155/2021/4917016

Zach. (2021a*). How to perform a Jarque-Bera test in Excel*. Statology. Retrieved from https://www.statology.org/jarque-bera-test-excel/

Zach. (2021b). *Standardization vs. normalization: What's the difference?* Statology. Retrieved from https://www.statology.org/standardization-vs-normalization/

Zach. (2021c). *The five assumptions for Pearson correlation*. Statology. Retrieved from https://www.statology.org/pearson-correlation-assumptions/

Zach. (2022). *Understanding the null hypothesis for ANOVA models*. Statology. Retrieved from https://www.statology.org/null-hypothesis-for-anova/

Appendix A

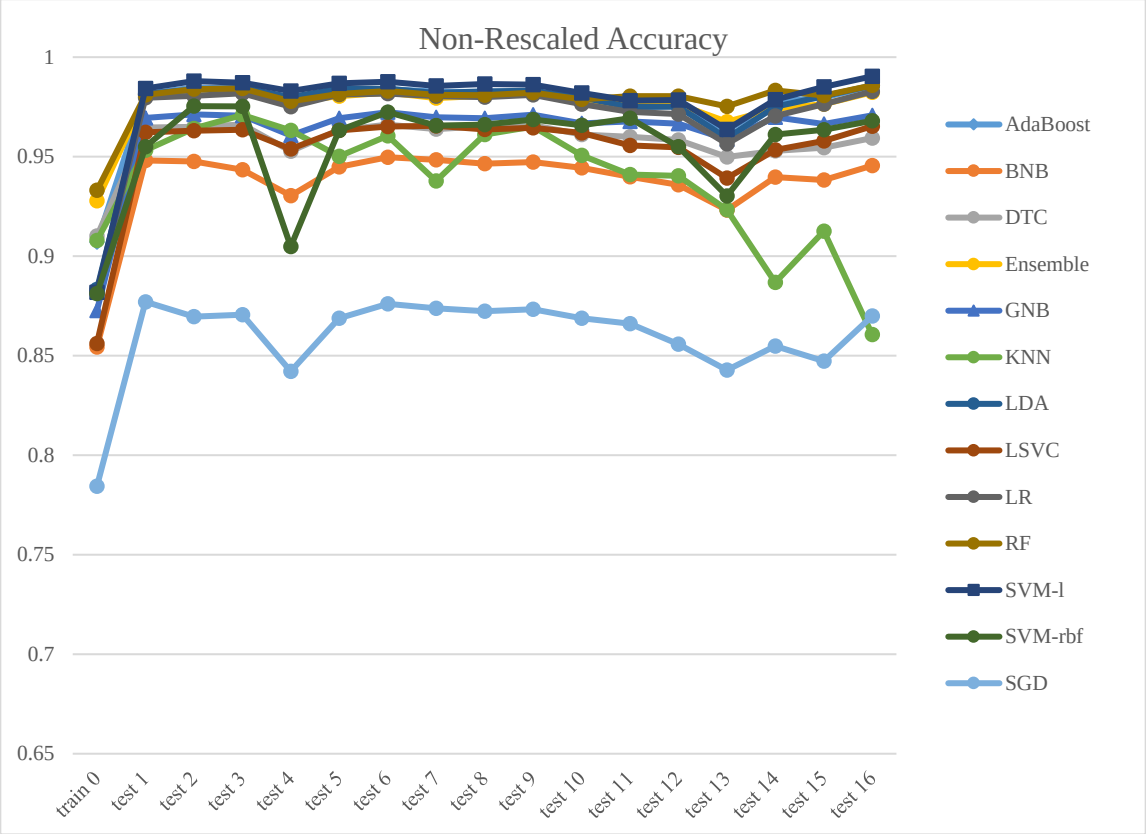


Figure 23: Non-Rescaled Accuracy Temporal Graph

Table 15: Non-Rescaled Accuracy Rank-Order

Model	train Accuracy	train rank	test1 Accuracy	test1 rank	test16 Accuracy	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.88189	7	0.98432	1	0.99038	1	0.10243	0.00606
RF	0.93307	1	0.98108	4	0.98583	2	0.04801	0.00475
LDA	0.88331	6	0.98146	3	0.98578	3	0.09814	0.00433
AdaBoost	0.90702	5	0.98161	2	0.98545	4	0.07460	0.00384
LR	0.88177	8	0.97962	6	0.98277	5	0.09785	0.00315
Ensemble	0.92786	2	0.98011	5	0.98227	6	0.05225	0.00216
GNB	0.87264	10	0.96951	7	0.97077	7	0.09688	0.00126
SVM-rbf	0.88114	9	0.95499	10	0.96809	8	0.07385	0.01310
LSVC	0.85618	11	0.96216	9	0.96528	9	0.10599	0.00312
DTC	0.91012	3	0.96458	8	0.95937	10	0.05446	-0.00521
BNB	0.85437	12	0.94809	12	0.94555	11	0.09372	-0.00254
SGD	0.78444	13	0.87708	13	0.86990	12	0.09265	-0.00719
KNN	0.90790	4	0.95330	11	0.86058	13	0.04540	-0.09273

Table 16: Non-Rescaled Accuracy Best Fit Line and MSE

Model	m Accuracy	m rank	b Accuracy	b rank	MSE Accuracy	MSE rank
SVM-rbf	0.00676	1	95.94041	11	3.09401	12
RF	-0.00433	2	98.16497	7	0.06188	1
GNB	-0.01816	3	96.97132	8	0.12611	3
Ensemble	-0.04391	4	98.24586	5	0.11270	2
AdaBoost	-0.04444	5	98.38447	3	0.23096	5
SVM-l	-0.04858	6	98.73537	1	0.32773	7
LSVC	-0.05580	7	96.43108	10	0.38307	9
LDA	-0.05668	8	98.45921	2	0.32569	6
BNB	-0.05698	9	94.69289	12	0.42918	10
LR	-0.06271	10	98.20886	6	0.33577	8
DTC	-0.06725	11	96.62736	9	0.17978	4
SGD	-0.10628	12	87.33667	13	1.06846	11
KNN	-0.51019	13	98.34838	4	3.30900	13

accuracy = (mx+b)/100

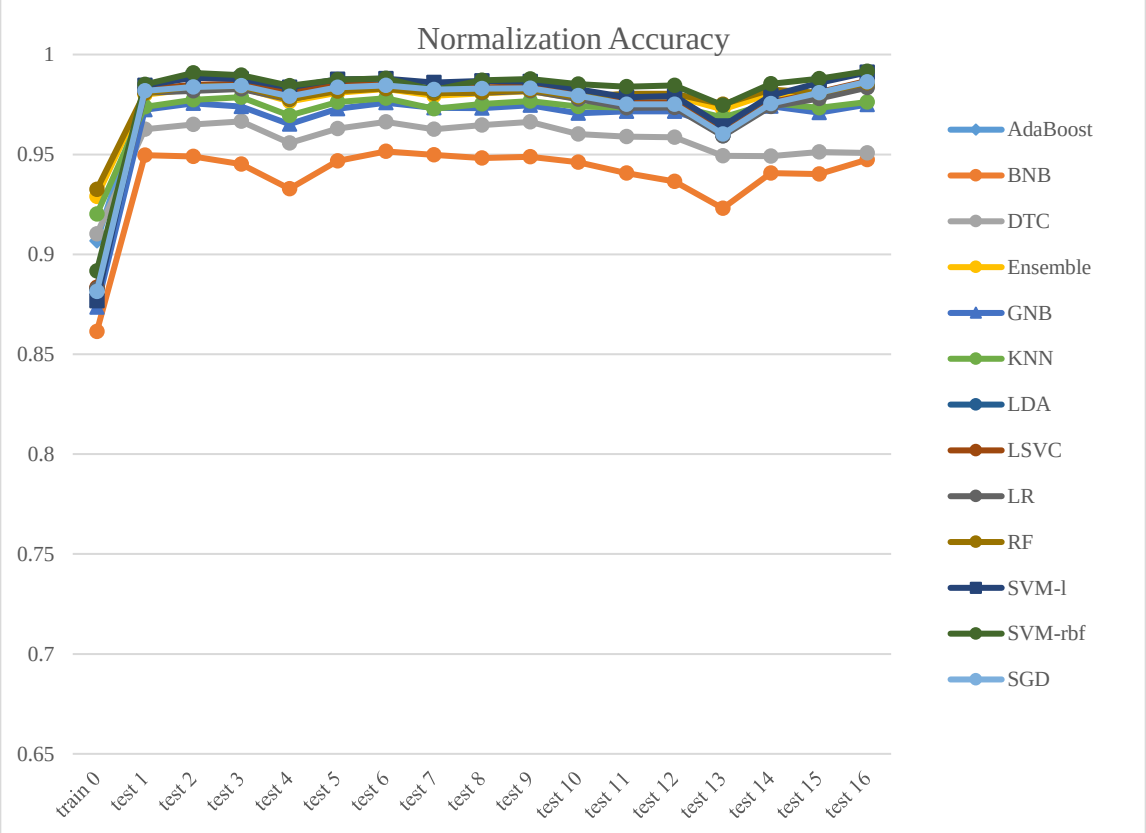


Figure 24: Normalized Accuracy Temporal Graph

Table 17: Normalized Accuracy Rank-Order

Model	train Accuracy	train rank	test1 Accuracy	test1 rank	test16 Accuracy	test16 rank	train to test1 delta	test1 to test16 delta
SVM-rbf	0.89170	6	0.98495	1	0.99164	1	0.09325	0.00669
SVM-l	0.87677	11	0.98459	2	0.99105	2	0.10782	0.00646
LSVC	0.88345	7	0.98263	3	0.98680	3	0.09919	0.00416
SGD	0.88135	10	0.98180	5	0.98635	4	0.10046	0.00454
RF	0.93250	1	0.98069	8	0.98576	5	0.04819	0.00507
LDA	0.88281	8	0.98146	6	0.98571	6	0.09865	0.00426
AdaBoost	0.90667	5	0.98189	4	0.98539	7	0.07522	0.00350
Ensemble	0.92901	2	0.97986	9	0.98422	8	0.05086	0.00435
LR	0.88267	9	0.98091	7	0.98341	9	0.09824	0.00251
KNN	0.92009	3	0.97386	10	0.97626	10	0.05377	0.00240
GNB	0.87358	12	0.97243	11	0.97494	11	0.09886	0.00251
DTC	0.91022	4	0.96259	12	0.95074	12	0.05237	-0.01185
BNB	0.86136	13	0.94958	13	0.94738	13	0.08822	-0.00220

Table 18: Normalized Accuracy Best Fit Line and MSE

Model	m Accuracy	m rank	b Accuracy	b rank	MSE Accuracy	MSE rank
RF	-0.01912	1	98.25817	8	0.06750	1
KNN	-0.02021	2	97.63464	10	0.07163	2
GNB	-0.02103	3	97.38405	11	0.11110	4
Ensemble	-0.02754	4	98.21711	9	0.07961	3
SVM-rbf	-0.03486	5	98.87174	2	0.15087	5
SGD	-0.05180	6	98.42807	7	0.31659	9
AdaBoost	-0.06765	7	98.59789	5	0.23175	7
SVM-l	-0.07719	8	99.01935	1	0.34621	12
LR	-0.08538	9	98.55219	6	0.29726	8
LDA	-0.08547	10	98.70750	4	0.34232	11
BNB	-0.08604	11	95.09868	13	0.47275	13
LSVC	-0.08626	12	98.80564	3	0.32719	10
DTC	-0.09640	13	96.86790	12	0.17600	6

accuracy = (mx+b)/100

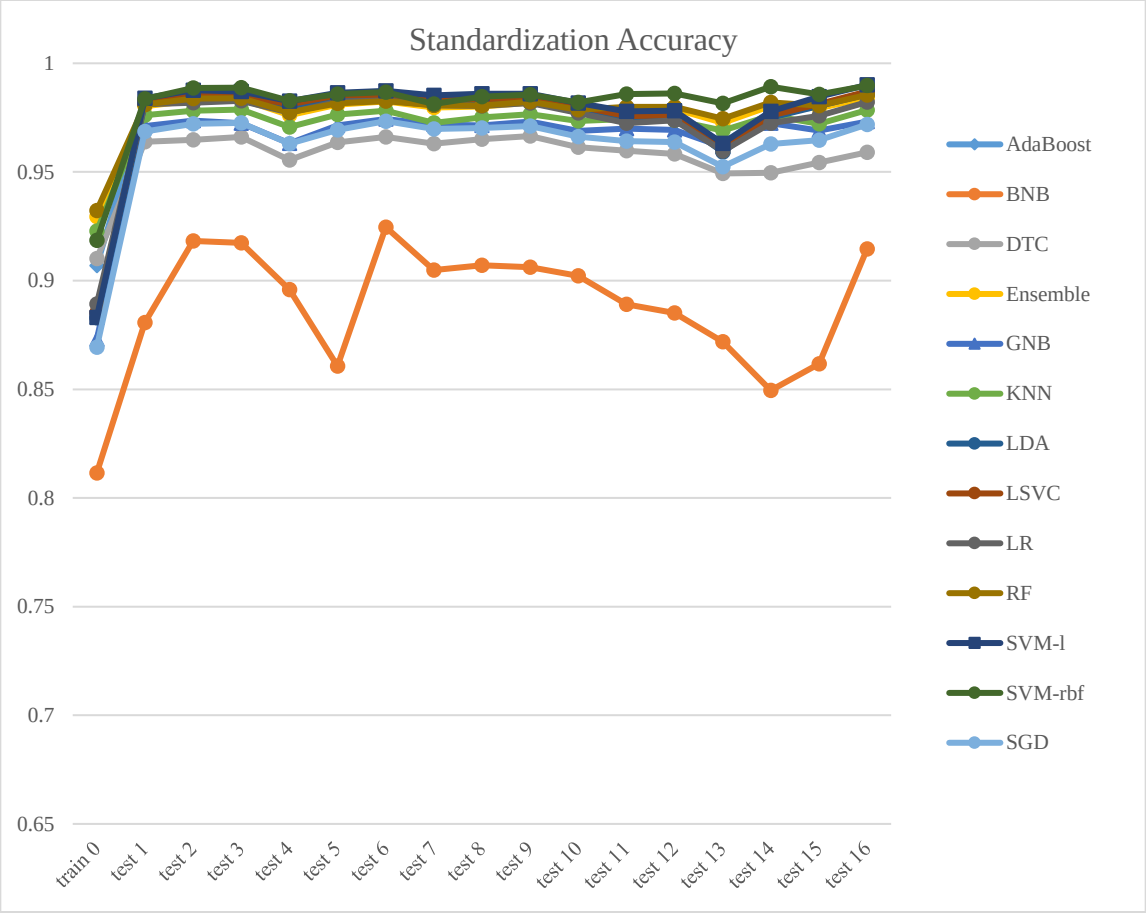


Figure 25: Standardized Accuracy Temporal Graph

Table 19: Standardized Accuracy Rank-Order

Model	train Accuracy	train rank	test1 Accuracy	test1 rank	test16 Accuracy	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.88309	9	0.98400	1	0.99005	1	0.10091	0.00606
SVM-rbf	0.91849	4	0.98359	2	0.98975	2	0.06510	0.00615
LSVC	0.88526	8	0.98288	3	0.98703	3	0.09762	0.00415
LDA	0.88286	10	0.98142	5	0.98572	4	0.09856	0.00430
RF	0.93230	1	0.98098	7	0.98546	5	0.04868	0.00448
AdaBoost	0.90692	6	0.98157	4	0.98538	6	0.07465	0.00381
Ensemble	0.92947	2	0.98019	8	0.98408	7	0.05072	0.00389
LR	0.88926	7	0.98107	6	0.98213	8	0.09181	0.00107
KNN	0.92289	3	0.97616	9	0.97854	9	0.05327	0.00238
GNB	0.87305	11	0.97111	10	0.97320	10	0.09806	0.00210
SGD	0.86939	12	0.96860	11	0.97183	11	0.09921	0.00323
DTC	0.91021	5	0.96385	12	0.95904	12	0.05364	-0.00481
BNB	0.81156	13	0.88076	13	0.91462	13	0.06920	0.03387

Table 20: Standardized Accuracy Best Fit Line and MSE

Model	m Accuracy	m rank	b Accuracy	b rank	MSE Accuracy	MSE rank
SVM-rbf	-0.00327	1	98.55156	6	0.07179	2
RF	-0.02145	2	98.25779	7	0.06824	1
GNB	-0.02346	3	97.23559	10	0.11836	5
Ensemble	-0.02792	4	98.22320	8	0.07961	4
KNN	-0.03198	5	97.76188	9	0.07646	3
SGD	-0.05091	6	97.15929	11	0.21667	7
AdaBoost	-0.06878	7	98.59329	4	0.23884	8
SVM-l	-0.08096	8	98.96791	1	0.35253	12
LDA	-0.08593	9	98.70227	3	0.34419	11
LSVC	-0.08712	10	98.82445	2	0.32857	10
DTC	-0.08864	11	96.84704	12	0.18297	6
LR	-0.09406	12	98.59326	5	0.26721	9
BNB	-0.27665	13	91.65676	13	4.48127	13

accuracy = (mx+b)/100

Table 21: Collective Accuracy Rank-Order

Type	Model	train Accuracy	train rank	test1 Accuracy	test1 rank	test16 Accuracy	test16 rank	train to test1 delta	test1 to test16 delta
Normalized	SVM-rbf	0.89170	17	0.98495	1	0.99164	1	0.09325	0.00669
Normalized	SVM-l	0.87677	30	0.98459	2	0.99105	2	0.10782	0.00646
Non-Rescaled	SVM-l	0.88189	26	0.98432	3	0.99038	3	0.10243	0.00606
Standardized	SVM-l	0.88309	22	0.98400	4	0.99005	4	0.10091	0.00606
Standardized	SVM-rbf	0.91849	9	0.98359	5	0.98975	5	0.06510	0.00615
Standardized	LSVC	0.88526	19	0.98288	6	0.98703	6	0.09762	0.00415
Normalized	LSVC	0.88345	20	0.98263	7	0.98680	7	0.09919	0.00416
Normalized	SGD	0.88135	28	0.98180	9	0.98635	8	0.10046	0.00454
Non-Rescaled	RF	0.93307	1	0.98108	15	0.98583	9	0.04801	0.00475
Non-Rescaled	LDA	0.88331	21	0.98146	12	0.98578	10	0.09814	0.00433
Normalized	RF	0.93250	2	0.98069	19	0.98576	11	0.04819	0.00507
Standardized	LDA	0.88286	23	0.98142	14	0.98572	12	0.09856	0.00430
Normalized	LDA	0.88281	24	0.98146	13	0.98571	13	0.09865	0.00426
Standardized	RF	0.93230	3	0.98098	17	0.98546	14	0.04868	0.00448
Non-Rescaled	AdaBoost	0.90702	14	0.98161	10	0.98545	15	0.07460	0.00384
Normalized	AdaBoost	0.90667	16	0.98189	8	0.98539	16	0.07522	0.00350
Standardized	AdaBoost	0.90692	15	0.98157	11	0.98538	17	0.07465	0.00381
Normalized	Ensemble	0.92901	5	0.97986	22	0.98422	18	0.05086	0.00435
Standardized	Ensemble	0.92947	4	0.98019	20	0.98408	19	0.05072	0.00389
Normalized	LR	0.88267	25	0.98091	18	0.98341	20	0.09824	0.00251
Non-Rescaled	LR	0.88177	27	0.97962	23	0.98277	21	0.09785	0.00315
Non-Rescaled	Ensemble	0.92786	6	0.98011	21	0.98227	22	0.05225	0.00216
Standardized	LR	0.88926	18	0.98107	16	0.98213	23	0.09181	0.00107
Standardized	KNN	0.92289	7	0.97616	24	0.97854	24	0.05327	0.00238
Normalized	KNN	0.92009	8	0.97386	25	0.97626	25	0.05377	0.00240
Normalized	GNB	0.87358	31	0.97243	26	0.97494	26	0.09886	0.00251
Standardized	GNB	0.87305	32	0.97111	27	0.97320	27	0.09806	0.00210
Standardized	SGD	0.86939	34	0.96860	29	0.97183	28	0.09921	0.00323
Non-Rescaled	GNB	0.87264	33	0.96951	28	0.97077	29	0.09688	0.00126
Non-Rescaled	SVM-rbf	0.88114	29	0.95499	34	0.96809	30	0.07385	0.01310
Non-Rescaled	LSVC	0.85618	36	0.96216	33	0.96528	31	0.10599	0.00312
Non-Rescaled	DTC	0.91012	12	0.96458	30	0.95937	32	0.05446	-0.00521
Standardized	DTC	0.91021	11	0.96385	31	0.95904	33	0.05364	-0.00481
Normalized	DTC	0.91022	10	0.96259	32	0.95074	34	0.05237	-0.01185
Normalized	BNB	0.86136	35	0.94958	36	0.94738	35	0.08822	-0.00220
Non-Rescaled	BNB	0.85437	37	0.94809	37	0.94555	36	0.09372	-0.00254
Standardized	BNB	0.81156	38	0.88076	38	0.91462	37	0.06920	0.03387

Non-Rescaled	SGD	0.78444	39	0.87708	39	0.86990	38	0.09265	-0.00719
Non-Rescaled	KNN	0.90790	13	0.95330	35	0.86058	39	0.04540	-0.09273

Table 22: Collective Accuracy Best Fit Line and MSE

Type	Model	m Accuracy	m rank	b Accuracy	b rank	MSE Accuracy	MSE rank
Non-Rescaled	SVM-rbf	0.00676	1	95.94041	35	3.09401	37
Standardized	SVM-rbf	-0.00327	2	98.55156	13	0.07179	5
Non-Rescaled	RF	-0.00433	3	98.16497	24	0.06188	1
Non-Rescaled	GNB	-0.01816	4	96.97132	30	0.12611	12
Normalized	RF	-0.01912	5	98.25817	18	0.06750	2
Normalized	KNN	-0.02021	6	97.63464	26	0.07163	4
Normalized	GNB	-0.02103	7	97.38405	27	0.11110	9
Standardized	RF	-0.02145	8	98.25779	19	0.06824	3
Standardized	GNB	-0.02346	9	97.23559	28	0.11836	11
Normalized	Ensemble	-0.02754	10	98.21711	22	0.07961	7
Standardized	Ensemble	-0.02792	11	98.22320	21	0.07961	8
Standardized	KNN	-0.03198	12	97.76188	25	0.07646	6
Normalized	SVM-rbf	-0.03486	13	98.87174	3	0.15087	13
Non-Rescaled	Ensemble	-0.04391	14	98.24586	20	0.11270	10
Non-Rescaled	AdaBoost	-0.04444	15	98.38447	16	0.23096	18
Non-Rescaled	SVM-l	-0.04858	16	98.73537	6	0.32773	26
Standardized	SGD	-0.05091	17	97.15929	29	0.21667	17
Normalized	SGD	-0.05180	18	98.42807	15	0.31659	23
Non-Rescaled	LSVC	-0.05580	19	96.43108	34	0.38307	33
Non-Rescaled	LDA	-0.05668	20	98.45921	14	0.32569	24
Non-Rescaled	BNB	-0.05698	21	94.69289	37	0.42918	34
Non-Rescaled	LR	-0.06271	22	98.20886	23	0.33577	28
Non-Rescaled	DTC	-0.06725	23	96.62736	33	0.17978	15
Normalized	AdaBoost	-0.06765	24	98.59789	9	0.23175	19
Standardized	AdaBoost	-0.06878	25	98.59329	10	0.23884	20
Normalized	SVM-l	-0.07719	26	99.01935	1	0.34621	31
Standardized	SVM-l	-0.08096	27	98.96791	2	0.35253	32
Normalized	LR	-0.08538	28	98.55219	12	0.29726	22
Normalized	LDA	-0.08547	29	98.70750	7	0.34232	29
Standardized	LDA	-0.08593	30	98.70227	8	0.34419	30
Normalized	BNB	-0.08604	31	95.09868	36	0.47275	35
Normalized	LSVC	-0.08626	32	98.80564	5	0.32719	25
Standardized	LSVC	-0.08712	33	98.82445	4	0.32857	27
Standardized	DTC	-0.08864	34	96.84704	32	0.18297	16
Standardized	LR	-0.09406	35	98.59326	11	0.26721	21
Normalized	DTC	-0.09640	36	96.86790	31	0.17600	14
Non-Rescaled	SGD	-0.10628	37	87.33667	39	1.06846	36
Standardized	BNB	-0.27665	38	91.65676	38	4.48127	39
Non-Rescaled	KNN	-0.51019	39	98.34838	17	3.30900	38

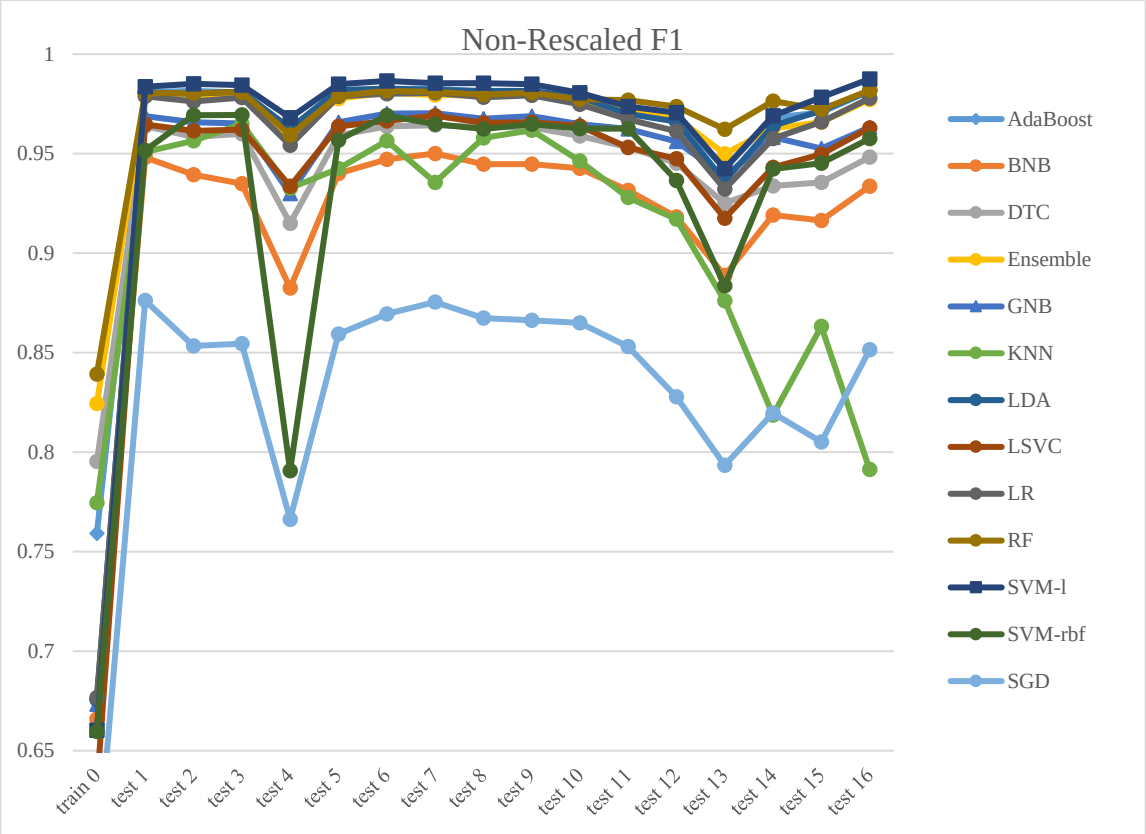


Figure 26: Non-Rescaled F1 Temporal Graph

Table 23: Non-Rescaled F1 Rank-Order

Model	train F1	train rank	test1 F1	test1 rank	test16 F1	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.66025	10	0.98361	1	0.98752503	1	0.32336	0.98752503
RF	0.83928	1	0.98040	4	0.98183038	2	0.14111	0.98183038
LDA	0.67564	7	0.98068	3	0.98168937	3	0.30504	0.98168937
AdaBoost	0.75911	5	0.98090	2	0.98127587	4	0.22178	0.98127587
LR	0.67664	6	0.97885	6	0.97787762	5	0.30220	0.97787762
Ensemble	0.82437	2	0.97937	5	0.97712659	6	0.15500	0.97712659
GNB	0.67333	8	0.96878	7	0.96325219	7	0.29544	0.96325219
LSVC	0.63144	12	0.96468	8	0.96279577	8	0.33324	0.96279577
SVM-rbf	0.65959	11	0.95156	10	0.95754529	9	0.29197	0.95754529
DTC	0.79527	3	0.96386	9	0.94825737	10	0.16859	0.94825737
BNB	0.66591	9	0.94805	12	0.93363745	11	0.28214	0.93363745
SGD	0.58935	13	0.87615	13	0.85145156	12	0.28680	0.85145156
KNN	0.77463	4	0.95079	11	0.79134263	13	0.17616	0.79134263

Table 24: Non-Rescaled F1 Best Fit Line and MSE

Model	m F1	m rank	b F1	b rank	MSE F1	MSE rank
SVM-rbf	0.01948	1	94.14059	11	19.64043	13
RF	-0.02651	2	97.87051	7	0.40955	1
GNB	-0.05476	3	96.51740	8	1.17335	5
AdaBoost	-0.08485	4	98.15979	4	0.97848	3
Ensemble	-0.08665	5	98.02409	5	0.64980	2
SVM-l	-0.08990	6	98.57759	2	1.08895	4
LDA	-0.10457	7	98.27711	3	1.18820	6
LSVC	-0.10727	8	96.47119	9	1.65502	8
LR	-0.11430	9	97.97895	6	1.37855	7
BNB	-0.12996	10	94.11617	12	3.54135	10
DTC	-0.13152	11	96.18424	10	1.88920	9
SGD	-0.21666	12	86.23813	13	8.81057	11
KNN	-0.88292	13	99.37690	1	9.93374	12

F1 = (mx+b)/100

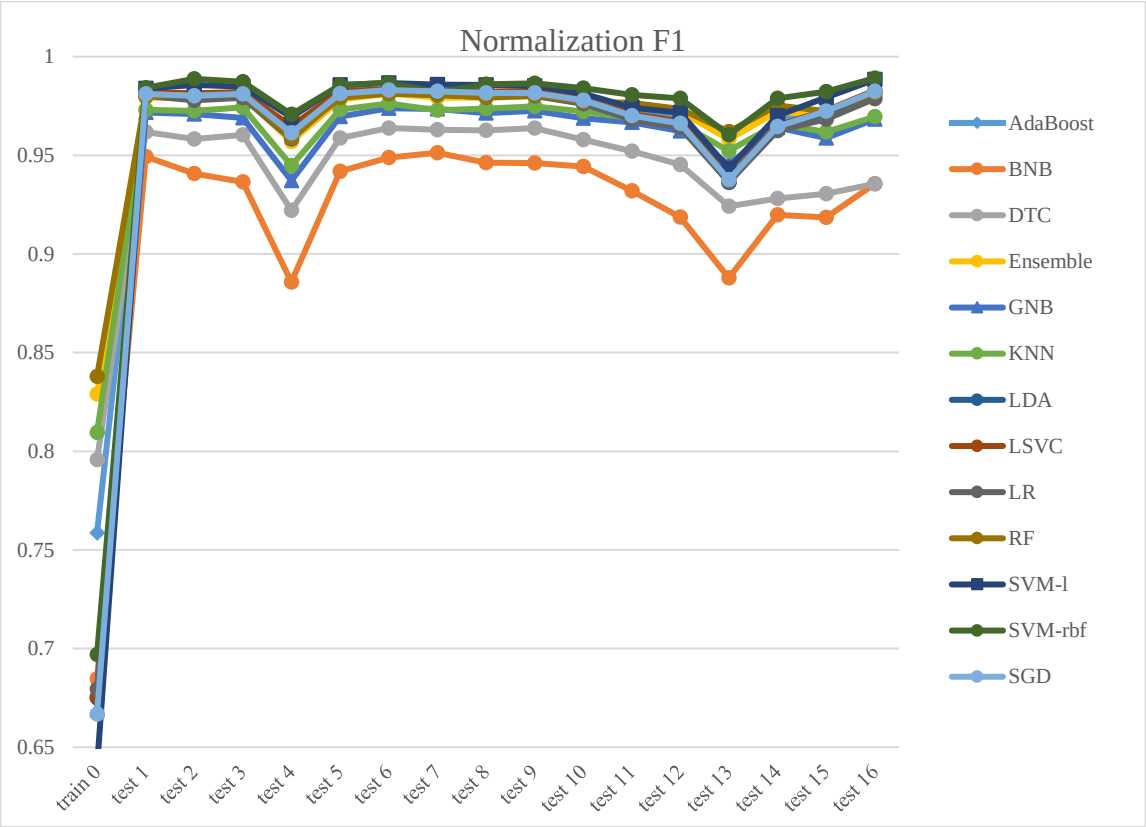


Figure 27: Normalized F1 Temporal Graph

Table 25: Normalized F1 Rank-Order

Model	train F1	train rank	test1 F1	test1 rank	test16 F1	test16 rank	train to test1 delta	test1 to test16 delta
SVM-rbf	0.69690	6	0.98427	1	0.98916	1	0.28737	0.00489
SVM-l	0.63975	13	0.98388	2	0.98838	2	0.34413	0.00449
LSVC	0.67558	9	0.98190	3	0.98298	3	0.30632	0.00108
SGD	0.66677	12	0.98107	5	0.98245	4	0.31430	0.00138
RF	0.83793	1	0.97999	8	0.98174	5	0.14207	0.00175
LDA	0.67497	10	0.98068	6	0.98160	6	0.30571	0.00092
AdaBoost	0.75861	5	0.98118	4	0.98120	7	0.22257	0.00001
Ensemble	0.82896	2	0.97915	9	0.97976	8	0.15019	0.00062
LR	0.67944	8	0.98018	7	0.97870	9	0.30074	-0.00147
KNN	0.80954	3	0.97303	10	0.96968	10	0.16349	-0.00334
GNB	0.66897	11	0.97168	11	0.96833	11	0.30271	-0.00335
DTC	0.79570	4	0.96177	12	0.93561	12	0.16607	-0.02616
BNB	0.68459	7	0.94941	13	0.93561	13	0.26482	-0.01380

Table 26: Normalized F1 Best Fit Line and MSE

Model	m F1	m rank	b F1	b rank	MSE F1	MSE rank
RF	-0.02764	1	97.86502	8	0.41587	1
Ensemble	-0.03954	2	97.81123	9	0.49649	3
SVM-rbf	-0.03957	3	98.53929	2	0.48044	2
GNB	-0.04172	4	96.88241	11	0.95158	6
KNN	-0.04459	5	97.20675	10	0.66437	4
AdaBoost	-0.08347	6	98.16999	6	0.94202	5
SVM-l	-0.08657	7	98.61465	1	1.06450	7
SGD	-0.09687	8	98.22889	5	1.17357	10
LSVC	-0.10333	9	98.38272	3	1.11000	8
LDA	-0.10425	10	98.27490	4	1.18552	11
LR	-0.10598	11	98.09813	7	1.13804	9
BNB	-0.13644	12	94.30850	13	3.54556	13
DTC	-0.18488	13	96.49773	12	1.58304	12

F1 = (mx+b)/100

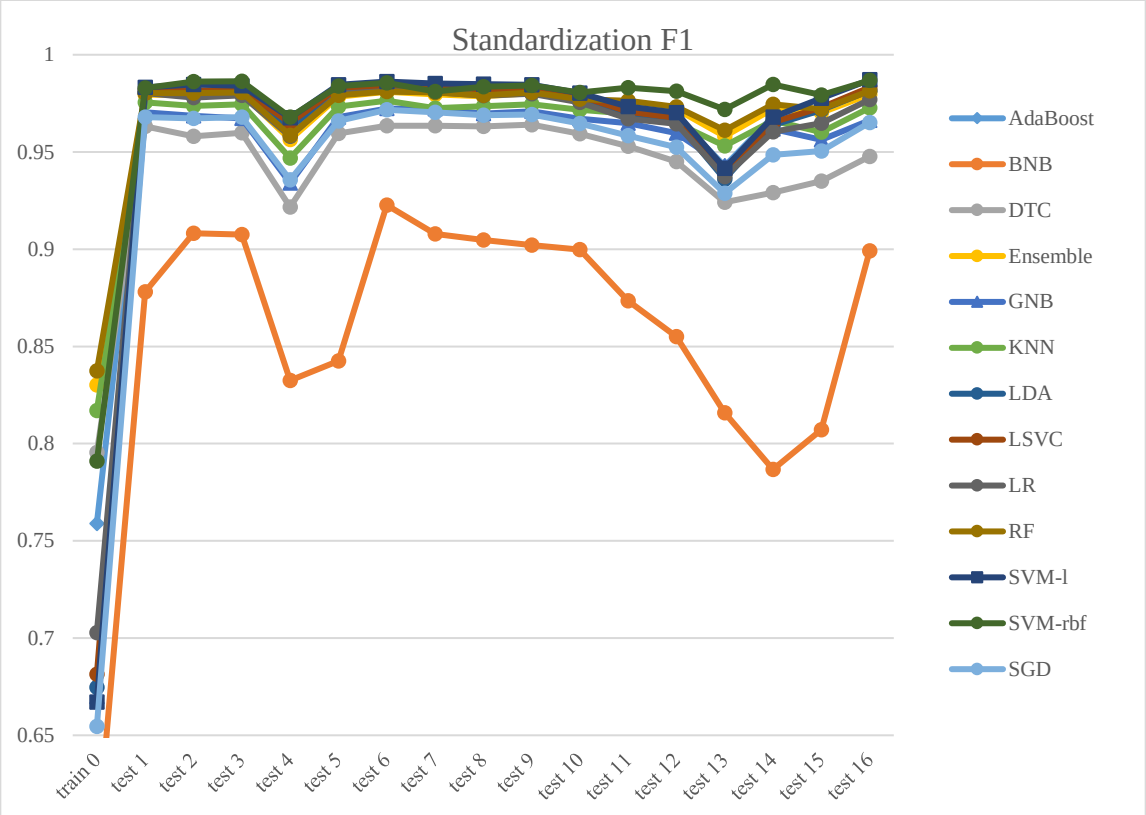


Figure 28: Standardized F1 Temporal Graph

Table 27: Standardized F1 Rank-Order

Model	train F1	train rank	test1 F1	test1 rank	test16 F1	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.66714	11	0.98328	1	0.98711	1	0.31614	0.00383
SVM-rbf	0.79099	5	0.98291	2	0.98678	2	0.19191	0.00387
LSVC	0.68142	8	0.98215	3	0.98327	3	0.30073	0.00112
LDA	0.67468	9	0.98065	5	0.98161	4	0.30596	0.00096
RF	0.83748	1	0.98030	7	0.98136	5	0.14282	0.00106
AdaBoost	0.75899	6	0.98085	4	0.98119	6	0.22187	0.00033
Ensemble	0.83017	2	0.97949	8	0.97961	7	0.14932	0.00012
LR	0.70285	7	0.98033	6	0.97706	8	0.27748	-0.00327
KNN	0.81709	3	0.97543	9	0.97267	9	0.15834	-0.00276
GNB	0.66992	10	0.97037	10	0.96623	10	0.30044	-0.00414
SGD	0.65453	12	0.96799	11	0.96511	11	0.31347	-0.00288
DTC	0.79543	4	0.96310	12	0.94773	12	0.16766	-0.01537
BNB	0.59302	13	0.87800	13	0.89914	13	0.28497	0.02115

Table 28: Standardized F1 Best Fit Line and MSE

Model	m F1	m rank	b F1	b rank	MSE F1	MSE rank
SVM-rbf	-0.00669	1	98.23858	4	0.24967	1
RF	-0.03049	2	97.86059	7	0.43304	2
Ensemble	-0.03995	3	97.81594	8	0.51173	3
GNB	-0.04584	4	96.71379	11	1.04405	6
KNN	-0.05083	5	97.30331	9	0.62766	4
AdaBoost	-0.08455	6	98.16189	5	0.95951	5
SVM-l	-0.09192	7	98.55095	1	1.10867	9
LSVC	-0.10347	8	98.39937	2	1.10109	8
LDA	-0.10452	9	98.26660	3	1.19007	10
SGD	-0.10475	10	96.84799	10	1.33843	11
LR	-0.11843	11	98.14875	6	1.07199	7
DTC	-0.14879	12	96.32695	12	1.66223	12
BNB	-0.39122	13	90.46906	13	13.81908	13

F1 = (mx+b)/100

Table 29: Collective F1 Rank-Order

Type	Model	train F1	train rank	test1 F1	test1 rank	test16 F1	test16 rank	train to test1 delta	test1 to test16 delta
Normalized	SVM-rbf	0.69690	18	0.98427	1	0.98916	1	0.28737	0.00489
Normalized	SVM-l	0.63975	36	0.98388	2	0.98838	2	0.34413	0.00449
Non-Rescaled	SVM-l	0.66025	33	0.98361	3	0.98752	3	0.32336	0.98752
Standardized	SVM-l	0.66714	30	0.98328	4	0.98711	4	0.31614	0.00383
Standardized	SVM-rbf	0.79099	12	0.98291	5	0.98678	5	0.19191	0.00387
Standardized	LSVC	0.68142	20	0.98215	6	0.98327	6	0.30073	0.00112
Normalized	LSVC	0.67558	24	0.98190	7	0.98298	7	0.30632	0.00108
Normalized	SGD	0.66677	31	0.98107	9	0.98245	8	0.31430	0.00138
Non-Rescaled	RF	0.83928	1	0.98040	15	0.98183	9	0.14111	0.98183
Normalized	RF	0.83793	2	0.97999	19	0.98174	10	0.14207	0.00175
Non-Rescaled	LDA	0.67564	23	0.98068	13	0.98168	11	0.30504	0.98168
Standardized	LDA	0.67468	26	0.98065	14	0.98161	12	0.30596	0.00096
Normalized	LDA	0.67497	25	0.98068	12	0.98160	13	0.30571	0.00092
Standardized	RF	0.83748	3	0.98030	17	0.98136	14	0.14282	0.00106
Non-Rescaled	AdaBoost	0.75911	14	0.98090	10	0.98127	15	0.22178	0.98127
Normalized	AdaBoost	0.75861	16	0.98118	8	0.98120	16	0.22257	0.00001
Standardized	AdaBoost	0.75899	15	0.98085	11	0.98119	17	0.22187	0.00033
Normalized	Ensemble	0.82896	5	0.97915	22	0.97976	18	0.15019	0.00062
Standardized	Ensemble	0.83017	4	0.97949	20	0.97961	19	0.14932	0.00012
Normalized	LR	0.67944	21	0.98018	18	0.97870	20	0.30074	-0.00147
Non-Rescaled	LR	0.67664	22	0.97885	23	0.97787	21	0.30220	0.97787
Non-Rescaled	Ensemble	0.82437	6	0.97937	21	0.97712	22	0.15500	0.97712
Standardized	LR	0.70285	17	0.98033	16	0.97706	23	0.27748	-0.00327
Standardized	KNN	0.81709	7	0.97543	24	0.97267	24	0.15834	-0.00276
Normalized	KNN	0.80954	8	0.97303	25	0.96968	25	0.16349	-0.00334
Normalized	GNB	0.66897	29	0.97168	26	0.96833	26	0.30271	-0.00335
Standardized	GNB	0.66992	28	0.97037	27	0.96623	27	0.30044	-0.00414
Standardized	SGD	0.65453	35	0.96799	29	0.96511	28	0.31347	-0.00288
Non-Rescaled	GNB	0.67333	27	0.96878	28	0.96325	29	0.29544	0.96325
Non-Rescaled	LSVC	0.63144	37	0.96468	30	0.96279	30	0.33324	0.96279
Non-Rescaled	SVM-rbf	0.65959	34	0.95156	34	0.95754	31	0.29197	0.95754
Non-Rescaled	DTC	0.79527	11	0.96386	31	0.94825	32	0.16859	0.94825
Standardized	DTC	0.79543	10	0.96310	32	0.94773	33	0.16766	-0.01537
Normalized	DTC	0.79570	9	0.96177	33	0.93561	34	0.16607	-0.02616
Normalized	BNB	0.68459	19	0.94941	36	0.93561	35	0.26482	-0.01380
Non-Rescaled	BNB	0.66591	32	0.94805	37	0.93363	36	0.28214	0.93363
Standardized	BNB	0.59302	38	0.87800	38	0.89914	37	0.28497	0.02115

Non-Rescaled	SGD	0.58935	39	0.87615	39	0.85145	38	0.28680	0.85145
Non-Rescaled	KNN	0.77463	13	0.95079	35	0.79134	39	0.17616	0.79134

Table 30: Collective F1 Best Fit Line and MSE

Type	Model	m F1	m rank	b F1	b rank	MSE F1	MSE rank
Non-Rescaled	SVM-rbf	0.01948	1	94.14059	36	19.64043	39
Standardized	SVM-rbf	-0.00669	2	98.23858	11	0.24967	1
Non-Rescaled	RF	-0.02651	3	97.87051	20	0.40955	2
Normalized	RF	-0.02764	4	97.86502	21	0.41587	3
Standardized	RF	-0.03049	5	97.86059	22	0.43304	4
Normalized	Ensemble	-0.03954	6	97.81123	24	0.49649	6
Normalized	SVM-rbf	-0.03957	7	98.53929	5	0.48044	5
Standardized	Ensemble	-0.03995	8	97.81594	23	0.51173	7
Normalized	GNB	-0.04172	9	96.88241	27	0.95158	12
Normalized	KNN	-0.04459	10	97.20675	26	0.66437	10
Standardized	GNB	-0.04584	11	96.71379	29	1.04405	15
Standardized	KNN	-0.05083	12	97.30331	25	0.62766	8
Non-Rescaled	GNB	-0.05476	13	96.51740	30	1.17335	23
Normalized	AdaBoost	-0.08347	14	98.16999	13	0.94202	11
Standardized	AdaBoost	-0.08455	15	98.16189	14	0.95951	13
Non-Rescaled	AdaBoost	-0.08485	16	98.15979	15	0.97848	14
Normalized	SVM-l	-0.08657	17	98.61465	2	1.06450	16
Non-Rescaled	Ensemble	-0.08665	18	98.02409	18	0.64980	9
Non-Rescaled	SVM-l	-0.08990	19	98.57759	3	1.08895	18
Standardized	SVM-l	-0.09192	20	98.55095	4	1.10867	20
Normalized	SGD	-0.09687	21	98.22889	12	1.17357	24
Normalized	LSVC	-0.10333	22	98.38272	7	1.11000	21
Standardized	LSVC	-0.10347	23	98.39937	6	1.10109	19
Normalized	LDA	-0.10425	24	98.27490	9	1.18552	25
Standardized	LDA	-0.10452	25	98.26660	10	1.19007	27
Non-Rescaled	LDA	-0.10457	26	98.27711	8	1.18820	26
Standardized	SGD	-0.10475	27	96.84799	28	1.33843	28
Normalized	LR	-0.10598	28	98.09813	17	1.13804	22
Non-Rescaled	LSVC	-0.10727	29	96.47119	32	1.65502	31
Non-Rescaled	LR	-0.11430	30	97.97895	19	1.37855	29
Standardized	LR	-0.11843	31	98.14875	16	1.07199	17
Non-Rescaled	BNB	-0.12996	32	94.11617	37	3.54135	34
Non-Rescaled	DTC	-0.13152	33	96.18424	34	1.88920	33
Normalized	BNB	-0.13644	34	94.30850	35	3.54556	35
Standardized	DTC	-0.14879	35	96.32695	33	1.66223	32
Normalized	DTC	-0.18488	36	96.49773	31	1.58304	30
Non-Rescaled	SGD	-0.21666	37	86.23813	39	8.81057	36
Standardized	BNB	-0.39122	38	90.46906	38	13.81908	38
Non-Rescaled	KNN	-0.88292	39	99.37690	1	9.93374	37

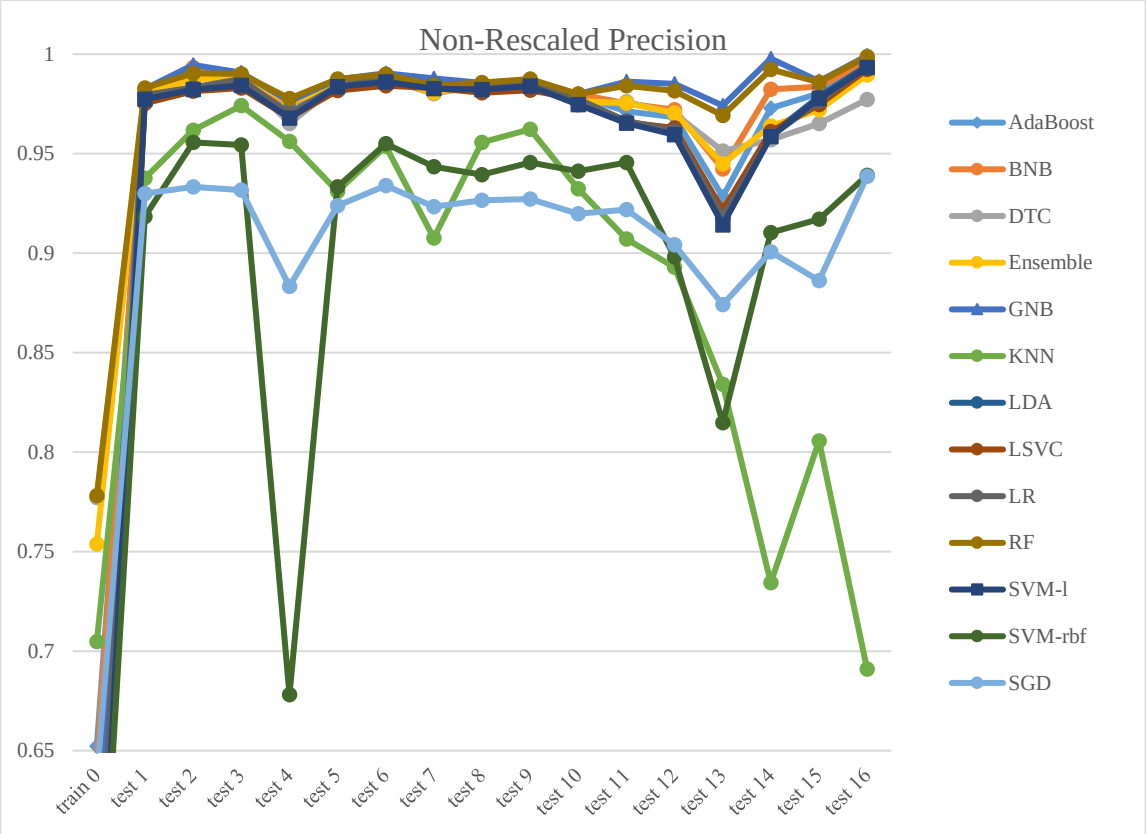


Figure 29: Non-Rescaled Precision Temporal Graph

Table 31: Non-Rescaled Precision Rank-Order

Model	train Precision	train rank	test1 Precision	test1 rank	test16 Precision	test16 rank	train to test1 delta	test1 to test16 delta
GNB	0.58449	8	0.98245	2	0.99919	1	0.39796	0.01674
RF	0.77806	1	0.98294	1	0.99879	2	0.20488	0.01585
BNB	0.64438	6	0.98230	3	0.99644	3	0.33792	0.01414
AdaBoost	0.65229	5	0.98068	6	0.99438	4	0.32839	0.01369
LDA	0.54111	10	0.97771	8	0.99422	5	0.43660	0.01651
LR	0.55078	9	0.97944	7	0.99341	6	0.42867	0.01397
SVM-l	0.51155	13	0.97741	9	0.99329	7	0.46587	0.01588
LSVC	0.53312	11	0.97526	10	0.99224	8	0.44214	0.01697
Ensemble	0.75386	3	0.98091	5	0.98951	9	0.22705	0.00860
DTC	0.77718	2	0.98107	4	0.97723	10	0.20389	-0.00384
SVM-rbf	0.51274	12	0.91837	13	0.93910	11	0.40563	0.02073
SGD	0.63576	7	0.92992	12	0.93858	12	0.29417	0.00865
KNN	0.70472	4	0.93764	11	0.69095	13	0.23292	-0.24669

Table 32: Non-Rescaled Precision Best Fit Line and MSE

Model	m Precision	m rank	b Precision	b rank	MSE Precision	MSE rank
SVM-rbf	0.03836	1	90.85228	12	47.46196	12
GNB	0.02127	2	98.51009	8	0.43297	2
RF	0.00668	3	98.48803	9	0.42307	1
BNB	-0.06479	4	98.67224	5	1.30673	5
LSVC	-0.10722	5	98.25017	11	2.23275	7
AdaBoost	-0.10767	6	98.76576	4	1.97571	6
Ensemble	-0.11630	7	98.78961	3	0.96118	4
LDA	-0.12653	8	98.53009	7	2.72070	9
SVM-l	-0.12973	9	98.43390	10	2.87915	10
LR	-0.13925	10	98.66458	6	2.62933	8
DTC	-0.14804	11	98.86103	2	0.76780	3
SGD	-0.21666	12	86.23813	13	55.37064	13
KNN	-1.45336	13	101.96160	1	23.86042	11

Precision = (mx+b)/100

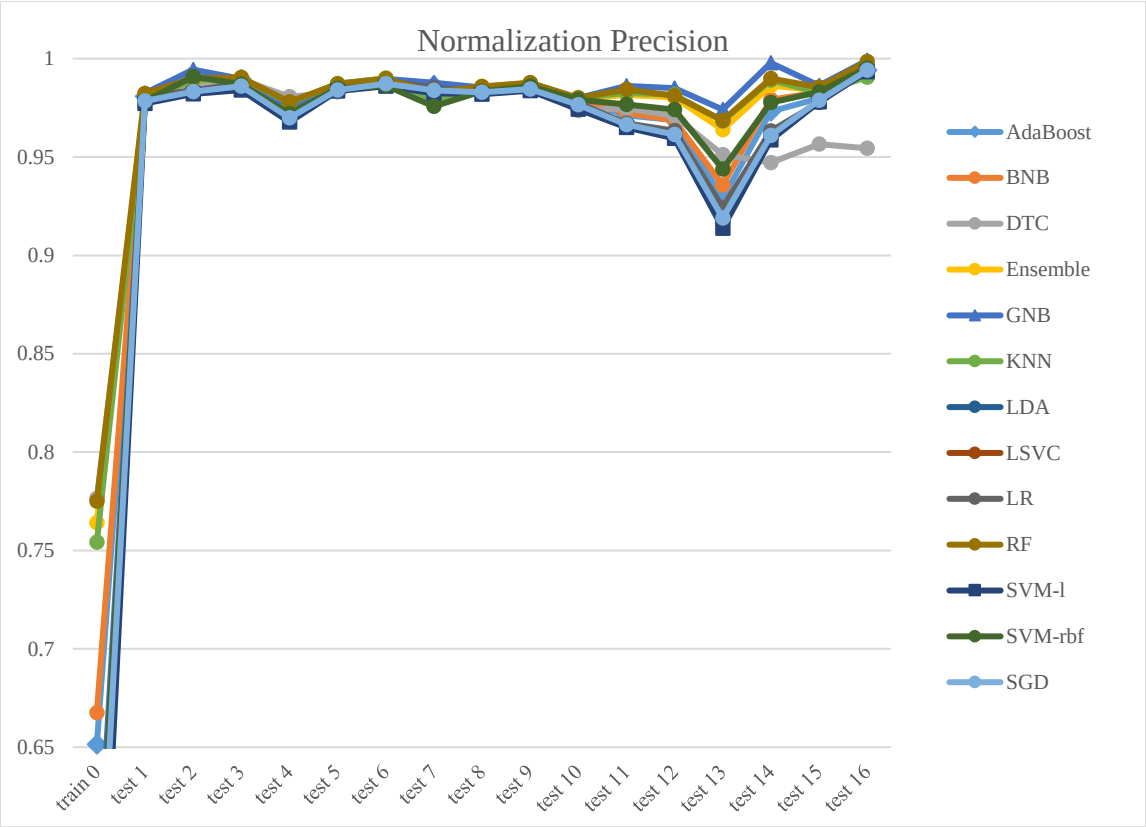


Figure 30: Normalized Precision Temporal Graph

Table 33: Normalized Precision Rank-Order

Model	train Precision	train rank	test1 Precision	test1 rank	test16 Precision	test16 rank	train to test1 delta	test1 to test16 delta
GNB	0.56756	7	0.98240	1	0.99908	1	0.41484	0.01669
RF	0.77493	2	0.98230	2	0.99852	2	0.20737	0.01622
Ensemble	0.76406	3	0.98199	3	0.99676	3	0.21793	0.01477
BNB	0.66754	5	0.98194	4	0.99594	4	0.31440	0.01399
SVM-rbf	0.55299	8	0.97837	10	0.99527	5	0.42538	0.01690
LSVC	0.53900	11	0.97848	9	0.99431	6	0.43948	0.01583
LR	0.55228	9	0.98066	6	0.99422	7	0.42838	0.01357
LDA	0.54042	10	0.97771	11	0.99421	8	0.43730	0.01649
AdaBoost	0.65135	6	0.98078	5	0.99416	9	0.32943	0.01338
SGD	0.53090	12	0.97849	8	0.99402	10	0.44759	0.01554
SVM-l	0.48669	13	0.97727	13	0.99330	11	0.49058	0.01603
KNN	0.75421	4	0.97935	7	0.99059	12	0.22514	0.01124
DTC	0.77648	1	0.97751	12	0.95442	13	0.20103	-0.02309

Table 34: Normalized Precision Best Fit Line and MSE

Model	m Precision	m rank	b Precision	b rank	MSE Precision	MSE rank
GNB	0.02419	1	98.46410	10	0.43493	3
RF	0.00263	2	98.48718	8	0.41961	2
KNN	-0.01163	3	98.47576	9	0.34295	1
Ensemble	-0.01545	4	98.52574	7	0.48968	4
SVM-rbf	-0.04638	5	98.35769	12	1.18262	6
BNB	-0.08202	6	98.68422	3	1.61711	7
SGD	-0.09687	7	98.22889	13	2.61030	10
AdaBoost	-0.10663	8	98.76113	2	1.92312	8
LR	-0.12466	9	98.67168	4	2.24653	9
LDA	-0.12631	10	98.53132	6	2.71782	12
LSVC	-0.12657	11	98.56090	5	2.62354	11
SVM-l	-0.12819	12	98.40941	11	2.89839	13
DTC	-0.23977	13	99.41631	1	0.60196	5

Precision = (mx+b)/100

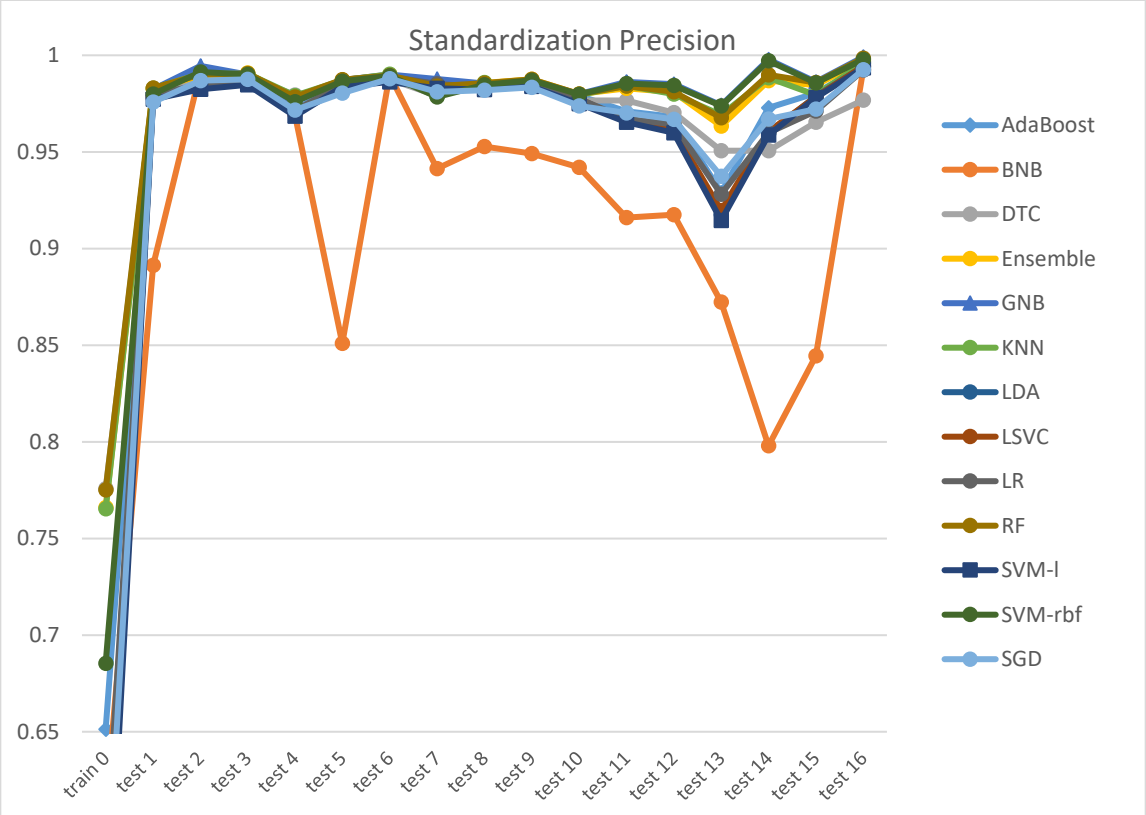


Figure 31: Standardized Precision Temporal Graph

Table 35: Standardized Precision Rank-Order

Model	train Precision	train rank	test1 Precision	test1 rank	test16 Precision	test16 rank	train to test1 delta	test1 to test16 delta
GNB	0.57263	9	0.98243	4	0.99912	1	0.40979	0.01669
RF	0.77518	2	0.98310	1	0.99855	2	0.20792	0.01546
SVM-rbf	0.68545	5	0.97993	8	0.99784	3	0.29448	0.01792
Ensemble	0.76605	3	0.98264	3	0.99726	4	0.21660	0.01461
KNN	0.76533	4	0.98294	2	0.99601	5	0.21761	0.01307
AdaBoost	0.65132	6	0.98080	5	0.99457	6	0.32947	0.01377
LSVC	0.54534	11	0.97853	9	0.99435	7	0.43319	0.01582
LDA	0.53980	12	0.97781	10	0.99421	8	0.43801	0.01640
SVM-l	0.52060	13	0.97722	11	0.99349	9	0.45662	0.01627
BNB	0.61007	7	0.89128	13	0.99274	10	0.28121	0.10145
SGD	0.55274	10	0.97577	12	0.99254	11	0.42303	0.01677
LR	0.58200	8	0.97997	6	0.99233	12	0.39797	0.01237
DTC	0.77577	1	0.97994	7	0.97675	13	0.20417	-0.00319

Table 36: Standardized Precision Best Fit Line and MSE

Model	m Precision	m rank	b Precision	b rank	MSE Precision	MSE rank
SVM-rbf	0.03297	1	98.26597	11	0.41687	2
GNB	0.02309	2	98.48192	9	0.43187	4
RF	-0.00137	3	98.53279	7	0.42899	3
Ensemble	-0.01505	4	98.56379	6	0.50549	5
KNN	-0.02063	5	98.58940	4	0.38589	1
SGD	-0.10475	6	96.84799	12	4.08209	12
AdaBoost	-0.10617	7	98.76015	3	1.95328	8
LSVC	-0.12547	8	98.57214	5	2.57564	9
LDA	-0.12626	9	98.53090	8	2.71295	10
SVM-l	-0.12862	10	98.44971	10	2.87297	11
LR	-0.14141	11	98.77649	2	1.95051	7
DTC	-0.17289	12	99.13055	1	0.72016	6
BNB	-0.48105	13	96.64212	13	28.42767	13

Precision = (mx+b)/100

Table 37: Collective Precision Rank-Order

Type	Model	train Precision	train rank	test1 Precision	test1 rank	test16 Precision	test16 rank	train to test1 delta	test1 to test16 delta
Non-Rescaled	GNB	0.58449	21	0.98245	5	0.99919	1	0.39796	0.01674
Standardized	GNB	0.57263	23	0.98243	6	0.99912	2	0.40979	0.01669
Normalized	GNB	0.56756	24	0.98240	7	0.99908	3	0.41484	0.01669
Non-Rescaled	RF	0.77806	1	0.98294	2	0.99879	4	0.20488	0.01585
Standardized	RF	0.77518	5	0.98310	1	0.99855	5	0.20792	0.01546
Normalized	RF	0.77493	6	0.98230	8	0.99852	6	0.20737	0.01622
Standardized	SVM-rbf	0.68545	13	0.97993	20	0.99784	7	0.29448	0.01792
Standardized	Ensemble	0.76605	7	0.98264	4	0.99726	8	0.21660	0.01461
Normalized	Ensemble	0.76406	9	0.98199	10	0.99676	9	0.21793	0.01477
Non-Rescaled	BNB	0.64438	18	0.98230	9	0.99644	10	0.33792	0.01414
Standardized	KNN	0.76533	8	0.98294	3	0.99601	11	0.21761	0.01307
Normalized	BNB	0.66754	14	0.98194	11	0.99594	12	0.31440	0.01399
Normalized	SVM-rbf	0.55299	25	0.97837	26	0.99527	13	0.42538	0.01690
Standardized	AdaBoost	0.65132	17	0.98080	14	0.99457	14	0.32947	0.01377
Non-Rescaled	AdaBoost	0.65229	15	0.98068	16	0.99438	15	0.32839	0.01369
Standardized	LSVC	0.54534	29	0.97853	23	0.99435	16	0.43319	0.01582
Normalized	LSVC	0.53900	33	0.97848	25	0.99431	17	0.43948	0.01583
Normalized	LR	0.55228	27	0.98066	17	0.99422	18	0.42838	0.01357
Non-Rescaled	LDA	0.54111	30	0.97771	29	0.99422	19	0.43660	0.01651
Standardized	LDA	0.53980	32	0.97781	27	0.99421	20	0.43801	0.01640
Normalized	LDA	0.54042	31	0.97771	28	0.99421	21	0.43730	0.01649
Normalized	AdaBoost	0.65135	16	0.98078	15	0.99416	22	0.32943	0.01338
Normalized	SGD	0.53090	35	0.97849	24	0.99402	23	0.44759	0.01554
Standardized	SVM-l	0.52060	36	0.97722	33	0.99349	24	0.45662	0.01627
Non-Rescaled	LR	0.55078	28	0.97944	21	0.99341	25	0.42867	0.01397
Normalized	SVM-l	0.48669	39	0.97727	32	0.99330	26	0.49058	0.01603
Non-Rescaled	SVM-l	0.51155	38	0.97741	31	0.99329	27	0.46587	0.01588
Standardized	BNB	0.61007	20	0.89128	39	0.99274	28	0.28121	0.10145
Standardized	SGD	0.55274	26	0.97577	34	0.99254	29	0.42303	0.01677
Standardized	LR	0.58200	22	0.97997	18	0.99233	30	0.39797	0.01237
Non-Rescaled	LSVC	0.53312	34	0.97526	35	0.99224	31	0.44214	0.01697
Normalized	KNN	0.75421	10	0.97935	22	0.99059	32	0.22514	0.01124
Non-Rescaled	Ensemble	0.75386	11	0.98091	13	0.98951	33	0.22705	0.00860
Non-Rescaled	DTC	0.77718	2	0.98107	12	0.97723	34	0.20389	-0.00384
Standardized	DTC	0.77577	4	0.97994	19	0.97675	35	0.20417	-0.00319
Normalized	DTC	0.77648	3	0.97751	30	0.95442	36	0.20103	-0.02309
Non-Rescaled	SVM-rbf	0.51274	37	0.91837	38	0.93910	37	0.40563	0.02073

Non-Rescaled	SGD	0.63576	19	0.92992	37	0.93858	38	0.29417	0.00865
Non-Rescaled	KNN	0.70472	12	0.93764	36	0.69095	39	0.23292	-0.24669

Table 38: Collective Precision Best Fit Line and MSE

Type	Model	m Precision	m rank	b Precision	b rank	MSE Precision	MSE rank
Non-Rescaled	SVM-rbf	0.03836	1	90.85228	38	47.46196	38
Standardized	SVM-rbf	0.03297	2	98.26597	33	0.41687	3
Normalized	GNB	0.02419	3	98.46410	28	0.43493	9
Standardized	GNB	0.02309	4	98.48192	26	0.43187	7
Non-Rescaled	GNB	0.02127	5	98.51009	23	0.43297	8
Non-Rescaled	RF	0.00668	6	98.48803	24	0.42307	5
Normalized	RF	0.00263	7	98.48718	25	0.41961	4
Standardized	RF	-0.00137	8	98.53279	18	0.42899	6
Normalized	KNN	-0.01163	9	98.47576	27	0.34295	1
Standardized	Ensemble	-0.01505	10	98.56379	16	0.50549	11
Normalized	Ensemble	-0.01545	11	98.52574	22	0.48968	10
Standardized	KNN	-0.02063	12	98.58940	14	0.38589	2
Normalized	SVM-rbf	-0.04638	13	98.35769	32	1.18262	16
Non-Rescaled	BNB	-0.06479	14	98.67224	11	1.30673	17
Normalized	BNB	-0.08202	15	98.68422	10	1.61711	18
Normalized	SGD	-0.09687	16	98.22889	35	2.61030	26
Standardized	SGD	-0.10475	17	96.84799	36	4.08209	35
Standardized	AdaBoost	-0.10617	18	98.76015	9	1.95328	21
Normalized	AdaBoost	-0.10663	19	98.76113	8	1.92312	19
Non-Rescaled	LSVC	-0.10722	20	98.25017	34	2.23275	23
Non-Rescaled	AdaBoost	-0.10767	21	98.76576	7	1.97571	22
Non-Rescaled	Ensemble	-0.11630	22	98.78961	5	0.96118	15
Normalized	LR	-0.12466	23	98.67168	12	2.24653	24
Standardized	LSVC	-0.12547	24	98.57214	15	2.57564	25
Standardized	LDA	-0.12626	25	98.53090	20	2.71295	29
Normalized	LDA	-0.12631	26	98.53132	19	2.71782	30
Non-Rescaled	LDA	-0.12653	27	98.53009	21	2.72070	31
Normalized	LSVC	-0.12657	28	98.56090	17	2.62354	27
Normalized	SVM-l	-0.12819	29	98.40941	31	2.89839	34
Standardized	SVM-l	-0.12862	30	98.44971	29	2.87297	32
Non-Rescaled	SVM-l	-0.12973	31	98.43390	30	2.87915	33
Non-Rescaled	LR	-0.13925	32	98.66458	13	2.62933	28
Standardized	LR	-0.14141	33	98.77649	6	1.95051	20
Non-Rescaled	DTC	-0.14804	34	98.86103	4	0.76780	14
Standardized	DTC	-0.17289	35	99.13055	3	0.72016	13
Non-Rescaled	SGD	-0.21666	36	86.23813	39	55.37064	39
Normalized	DTC	-0.23977	37	99.41631	2	0.60196	12
Standardized	BNB	-0.48105	38	96.64212	37	28.42767	37
Non-Rescaled	KNN	-1.45336	39	101.96160	1	23.86042	36

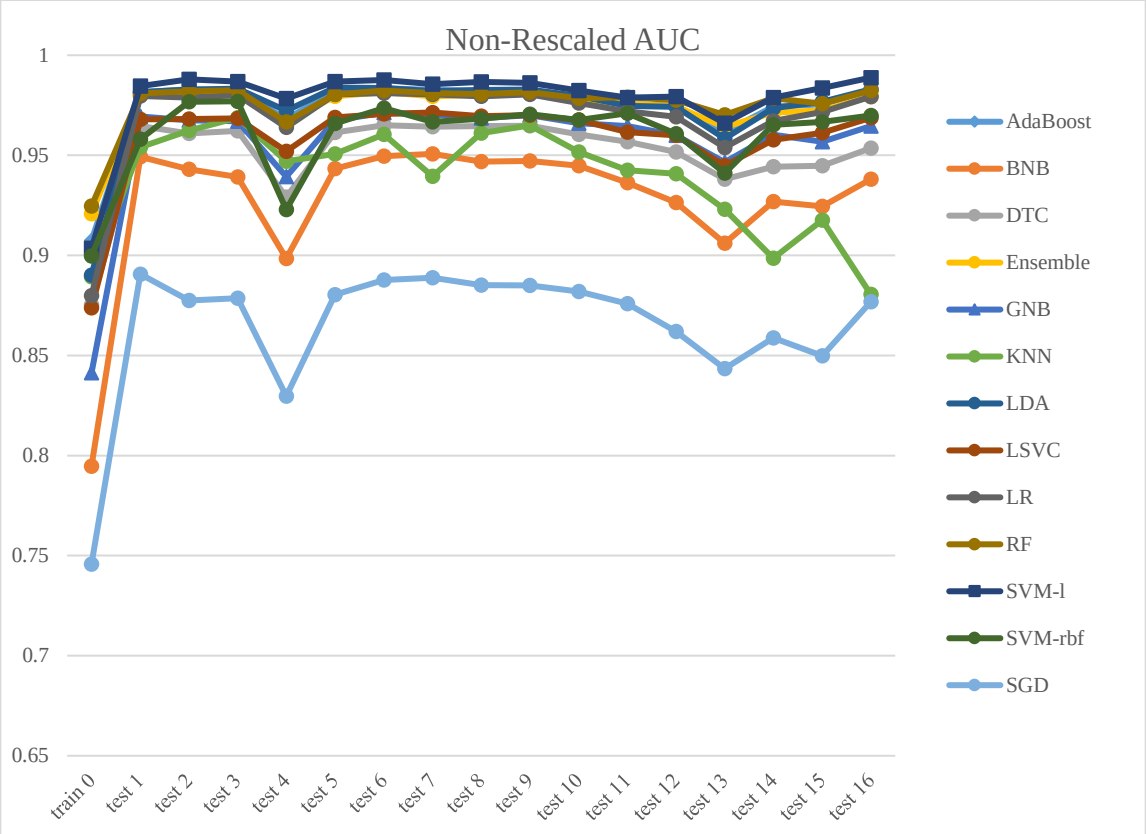


Figure 32: Non-Rescaled AUC Temporal Graph

Table 39: Non-Rescaled AUC Rank-Order

Model	train AUC	train rank	test1 AUC	test1 rank	test16 AUC	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.90345	4	0.98458	1	0.98881	1	0.08113	0.00423
LDA	0.89009	6	0.98155	3	0.98291	2	0.09146	0.00136
AdaBoost	0.90734	3	0.98159	2	0.98248	3	0.07425	0.00089
RF	0.92467	1	0.98098	4	0.98234	4	0.05631	0.00135
LR	0.87989	8	0.97958	6	0.97932	5	0.09969	-0.00025
Ensemble	0.92078	2	0.98004	5	0.97928	6	0.05926	-0.00075
SVM-rbf	0.89977	5	0.95804	10	0.96990	7	0.05827	0.01187
LSVC	0.87377	10	0.96795	8	0.96873	8	0.09417	0.00078
GNB	0.84126	11	0.96938	7	0.96464	9	0.12812	-0.00473
DTC	0.87528	9	0.96454	9	0.95360	10	0.08927	-0.01094
BNB	0.79470	12	0.94931	12	0.93806	11	0.15461	-0.01125
KNN	0.88936	7	0.95400	11	0.88047	12	0.06464	-0.07353
SGD	0.74575	13	0.89063	13	0.87688	13	0.14488	-0.01375

Table 40: Non-Rescaled AUC Best Fit Line and MSE

Model	m AUC	m rank	b AUC	b rank	MSE AUC	MSE rank
SVM-rbf	0.00543	1	96.34107	10	1.78522	10
RF	-0.01721	2	98.02146	6	0.18148	1
SVM-l	-0.04366	3	98.67217	1	0.27035	3
GNB	-0.04660	4	96.70798	9	0.70414	8
Ensemble	-0.04777	5	98.07055	4	0.23255	2
AdaBoost	-0.04860	6	98.23330	3	0.31771	4
LSVC	-0.05598	7	96.90469	8	0.47436	7
LDA	-0.06031	8	98.35766	2	0.34880	5
LR	-0.06879	9	98.04601	5	0.47011	6
DTC	-0.08968	10	96.30202	11	0.96000	9
BNB	-0.09994	11	94.41393	12	2.06082	11
SGD	-0.12754	12	88.28432	13	2.70248	13
KNN	-0.41020	13	97.63314	7	2.24576	12

AUC = (mx+b)/100

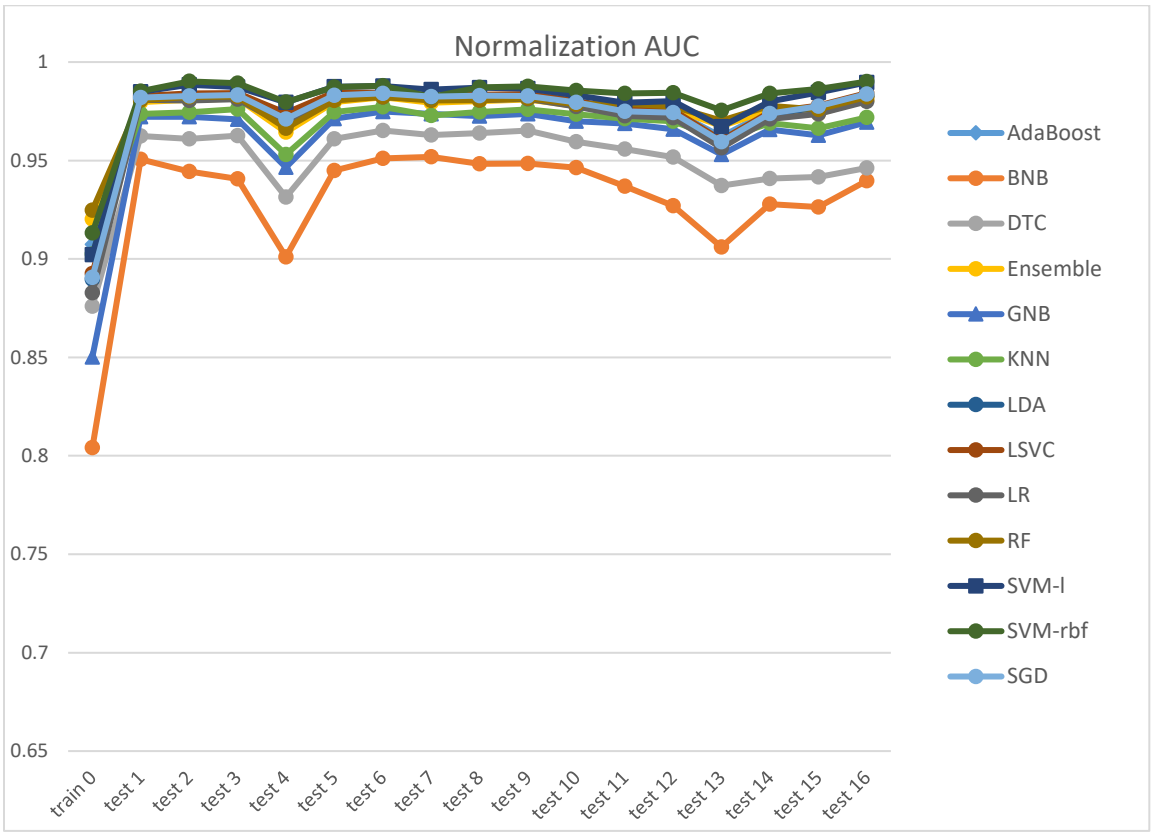


Figure 33: Normalized AUC Temporal Graph

Table 41: Normalized AUC Rank-Order

Model	train AUC	train rank	test1 AUC	test1 rank	test16 AUC	test16 rank	train to test1 delta	test1 to test16 delta
SVM-rbf	0.91312	3	0.98519	1	0.99008	1	0.07207	0.00489
SVM-l	0.90209	6	0.98488	2	0.98966	2	0.08279	0.00478
LSVC	0.89260	7	0.98275	3	0.98416	3	0.09014	0.00141
SGD	0.89066	8	0.98192	4	0.98374	4	0.09126	0.00182
LDA	0.88964	9	0.98155	6	0.98283	5	0.09191	0.00128
AdaBoost	0.90732	4	0.98188	5	0.98244	6	0.07456	0.00056
RF	0.92475	1	0.98060	8	0.98229	7	0.05585	0.00169
Ensemble	0.92020	2	0.97976	9	0.98065	8	0.05955	0.00089
LR	0.88275	10	0.98087	7	0.98000	9	0.09811	-0.00087
KNN	0.90250	5	0.97370	10	0.97184	10	0.07119	-0.00186
GNB	0.85019	12	0.97227	11	0.96943	11	0.12208	-0.00283
DTC	0.87596	11	0.96252	12	0.94624	12	0.08655	-0.01628
BNB	0.80411	13	0.95056	13	0.93977	13	0.14645	-0.01079

Table 42: Normalized AUC Best Fit Line and MSE

Model	m AUC	m rank	b AUC	b rank	MSE AUC	MSE rank
SVM-rbf	-0.01511	1	98.67026	2	0.62955	11
RF	-0.01707	2	98.00999	8	0.18701	1
Ensemble	-0.02476	3	97.94379	9	0.22468	2
KNN	-0.02967	4	97.34573	10	0.35745	7
GNB	-0.03510	5	97.06934	11	0.54666	9
SVM-l	-0.04037	6	98.71241	1	0.25641	3
AdaBoost	-0.04748	7	98.24451	6	0.30110	4
SGD	-0.05363	8	98.31778	5	0.60926	10
LSVC	-0.05950	9	98.46292	3	0.31530	5
LDA	-0.06004	10	98.35540	4	0.34752	6
LR	-0.06465	11	98.17318	7	0.37659	8
BNB	-0.10361	12	94.57666	13	2.03854	13
DTC	-0.11779	13	96.43306	12	0.88391	12

AUC = (mx+b)/100

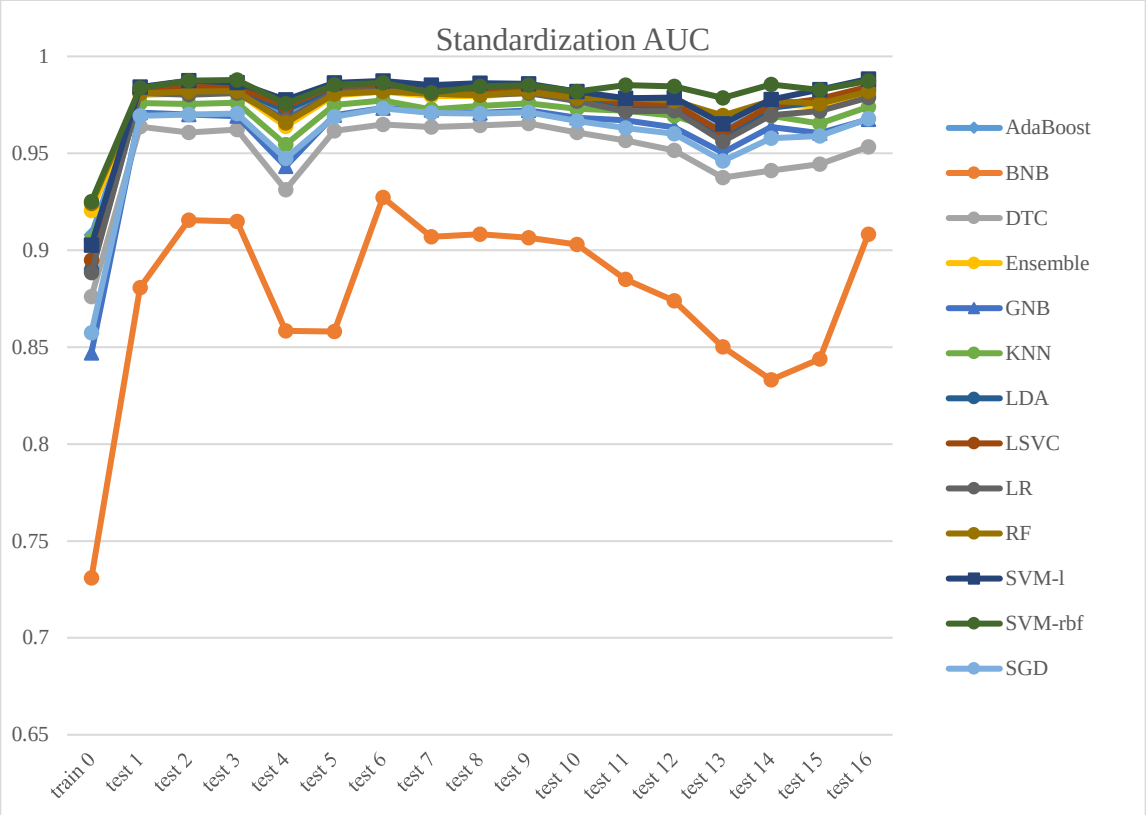


Figure 34: Standardized AUC Temporal Graph

Table 43: Standardized AUC Rank-Order

Model	train AUC	train rank	test1 AUC	test1 rank	test16 AUC	test16 rank	train to test1 delta	test1 to test16 delta
SVM-l	0.90260	6	0.98425	1	0.98837	1	0.08165	0.00411
SVM-rbf	0.92517	1	0.98369	2	0.98730	2	0.05852	0.00361
LSVC	0.89495	7	0.98300	3	0.98444	3	0.08805	0.00144
LDA	0.88994	8	0.98151	5	0.98283	4	0.09157	0.00133
AdaBoost	0.90796	4	0.98155	4	0.98237	5	0.07358	0.00082
RF	0.92405	2	0.98087	7	0.98192	6	0.05682	0.00104
Ensemble	0.92052	3	0.98008	8	0.98042	7	0.05956	0.00034
LR	0.88839	9	0.98105	6	0.97869	8	0.09266	-0.00236
KNN	0.90533	5	0.97599	9	0.97392	9	0.07066	-0.00207
SGD	0.85733	11	0.96928	11	0.96793	10	0.11195	-0.00135
GNB	0.84705	12	0.97096	10	0.96745	11	0.12391	-0.00351
DTC	0.87598	10	0.96380	12	0.95331	12	0.08781	-0.01049
BNB	0.73093	13	0.88064	13	0.90825	13	0.14971	0.02761

Table 44: Standardized AUC Best Fit Line and MSE

Model	m AUC	m rank	b AUC	b rank	MSE AUC	MSE rank
SVM-rbf	-0.00240	1	98.41493	3	0.10195	1
RF	-0.01941	2	98.00111	7	0.19740	2
Ensemble	-0.02559	3	97.94739	8	0.23257	3
KNN	-0.03497	4	97.42891	9	0.33569	7
GNB	-0.03863	5	96.90344	11	0.61172	10
SVM-l	-0.04605	6	98.64220	1	0.28321	4
AdaBoost	-0.04868	7	98.23684	5	0.30979	5
LSVC	-0.06007	8	98.47896	2	0.31715	6
LDA	-0.06038	9	98.34753	4	0.35073	8
SGD	-0.06500	10	97.00367	10	3.81516	12
LR	-0.07357	11	98.20092	6	0.36804	9
DTC	-0.09805	12	96.35188	12	0.92450	11
BNB	-0.23608	13	90.59171	13	7.01272	13

Table 45: Collective AUC Rank-Order

Type	Model	train AUC	train rank	test1 AUC	test1 rank	test16 AUC	test16 rank	train to test1 delta	test1 to test16 delta
Normalized	SVM-rbf	0.91312	8	0.98519	1	0.99008	1	0.07207	0.00489
Normalized	SVM-l	0.90209	16	0.98488	2	0.98966	2	0.08279	0.00478
Non-Rescaled	SVM-l	0.90345	13	0.98458	3	0.98881	3	0.08113	0.00423
Standardized	SVM-l	0.90260	14	0.98425	4	0.98837	4	0.08165	0.00411
Standardized	SVM-rbf	0.92517	1	0.98369	5	0.98730	5	0.05852	0.00361
Standardized	LSVC	0.89495	18	0.98300	6	0.98444	6	0.08805	0.00144
Normalized	LSVC	0.89260	19	0.98275	7	0.98416	7	0.09014	0.00141
Normalized	SGD	0.89066	20	0.98192	8	0.98374	8	0.09126	0.00182
Non-Rescaled	LDA	0.89009	21	0.98155	12	0.98291	9	0.09146	0.00136
Standardized	LDA	0.88994	22	0.98151	14	0.98283	10	0.09157	0.00133
Normalized	LDA	0.88964	23	0.98155	11	0.98283	11	0.09191	0.00128
Non-Rescaled	AdaBoost	0.90734	10	0.98159	10	0.98248	12	0.07425	0.00089
Normalized	AdaBoost	0.90732	11	0.98188	9	0.98244	13	0.07456	0.00056
Standardized	AdaBoost	0.90796	9	0.98155	13	0.98237	14	0.07358	0.00082
Non-Rescaled	RF	0.92467	3	0.98098	16	0.98234	15	0.05631	0.00135
Normalized	RF	0.92475	2	0.98060	19	0.98229	16	0.05585	0.00169
Standardized	RF	0.92405	4	0.98087	17	0.98192	17	0.05682	0.00104
Normalized	Ensemble	0.92020	7	0.97976	22	0.98065	18	0.05955	0.00089
Standardized	Ensemble	0.92052	6	0.98008	20	0.98042	19	0.05956	0.00034
Normalized	LR	0.88275	26	0.98087	18	0.98000	20	0.09811	-0.00087
Non-Rescaled	LR	0.87989	27	0.97958	23	0.97932	21	0.09969	-0.00025
Non-Rescaled	Ensemble	0.92078	5	0.98004	21	0.97928	22	0.05926	-0.00075
Standardized	LR	0.88839	25	0.98105	15	0.97869	23	0.09266	-0.00236
Standardized	KNN	0.90533	12	0.97599	24	0.97392	24	0.07066	-0.00207
Normalized	KNN	0.90250	15	0.97370	25	0.97184	25	0.07119	-0.00186
Non-Rescaled	SVM-rbf	0.89977	17	0.95804	34	0.96990	26	0.05827	0.01187
Normalized	GNB	0.85019	33	0.97227	26	0.96943	27	0.12208	-0.00283
Non-Rescaled	LSVC	0.87377	31	0.96795	30	0.96873	28	0.09417	0.00078
Standardized	SGD	0.85733	32	0.96928	29	0.96793	29	0.11195	-0.00135
Standardized	GNB	0.84705	34	0.97096	27	0.96745	30	0.12391	-0.00351
Non-Rescaled	GNB	0.84126	35	0.96938	28	0.96464	31	0.12812	-0.00473
Non-Rescaled	DTC	0.87528	30	0.96454	31	0.95360	32	0.08927	-0.01094
Standardized	DTC	0.87598	28	0.96380	32	0.95331	33	0.08781	-0.01049
Normalized	DTC	0.87596	29	0.96252	33	0.94624	34	0.08655	-0.01628
Normalized	BNB	0.80411	36	0.95056	36	0.93977	35	0.14645	-0.01079
Non-Rescaled	BNB	0.79470	37	0.94931	37	0.93806	36	0.15461	-0.01125
Standardized	BNB	0.73093	39	0.88064	39	0.90825	37	0.14971	0.02761

Non-Rescaled	KNN	0.88936	24	0.95400	35	0.88047	38	0.06464	-0.07353
Non-Rescaled	SGD	0.74575	38	0.89063	38	0.87688	39	0.14488	-0.01375

Table 46: Collective AUC Best Fit Line and MSE

Type	Model	m AUC	m rank	b AUC	b rank	MSE AUC	MSE rank
Non-Rescaled	SVM-rbf	0.00543	1	96.34107	34	1.78522	33
Standardized	SVM-rbf	-0.00240	2	98.41493	7	0.10195	1
Normalized	SVM-rbf	-0.01511	3	98.67026	3	0.62955	28
Normalized	RF	-0.01707	4	98.00999	20	0.18701	3
Non-Rescaled	RF	-0.01721	5	98.02146	19	0.18148	2
Standardized	RF	-0.01941	6	98.00111	21	0.19740	4
Normalized	Ensemble	-0.02476	7	97.94379	23	0.22468	5
Standardized	Ensemble	-0.02559	8	97.94739	22	0.23257	7
Normalized	KNN	-0.02967	9	97.34573	26	0.35745	20
Standardized	KNN	-0.03497	10	97.42891	25	0.33569	16
Normalized	GNB	-0.03510	11	97.06934	27	0.54666	25
Standardized	GNB	-0.03863	12	96.90344	30	0.61172	27
Normalized	SVM-l	-0.04037	13	98.71241	1	0.25641	8
Non-Rescaled	SVM-l	-0.04366	14	98.67217	2	0.27035	9
Standardized	SVM-l	-0.04605	15	98.64220	4	0.28321	10
Non-Rescaled	GNB	-0.04660	16	96.70798	31	0.70414	29
Normalized	AdaBoost	-0.04748	17	98.24451	12	0.30110	11
Non-Rescaled	Ensemble	-0.04777	18	98.07055	17	0.23255	6
Non-Rescaled	AdaBoost	-0.04860	19	98.23330	14	0.31771	15
Standardized	AdaBoost	-0.04868	20	98.23684	13	0.30979	12
Normalized	SGD	-0.05363	21	98.31778	11	0.60926	26
Non-Rescaled	LSVC	-0.05598	22	96.90469	29	0.47436	24
Normalized	LSVC	-0.05950	23	98.46292	6	0.31530	13
Normalized	LDA	-0.06004	24	98.35540	9	0.34752	17
Standardized	LSVC	-0.06007	25	98.47896	5	0.31715	14
Non-Rescaled	LDA	-0.06031	26	98.35766	8	0.34880	18
Standardized	LDA	-0.06038	27	98.34753	10	0.35073	19
Normalized	LR	-0.06465	28	98.17318	16	0.37659	22
Standardized	SGD	-0.06500	29	97.00367	28	3.81516	38
Non-Rescaled	LR	-0.06879	30	98.04601	18	0.47011	23
Standardized	LR	-0.07357	31	98.20092	15	0.36804	21
Non-Rescaled	DTC	-0.08968	32	96.30202	35	0.96000	32
Standardized	DTC	-0.09805	33	96.35188	33	0.92450	31
Non-Rescaled	BNB	-0.09994	34	94.41393	37	2.06082	35
Normalized	BNB	-0.10361	35	94.57666	36	2.03854	34
Normalized	DTC	-0.11779	36	96.43306	32	0.88391	30
Non-Rescaled	SGD	-0.12754	37	88.28432	39	2.70248	37
Standardized	BNB	-0.23608	38	90.59171	38	7.01272	39
Non-Rescaled	KNN	-0.41020	39	97.63314	24	2.24576	36

Appendix B

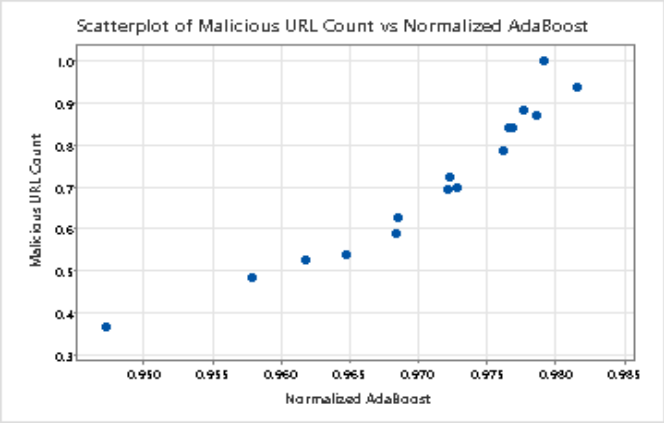


Figure 35: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized AdaBoost

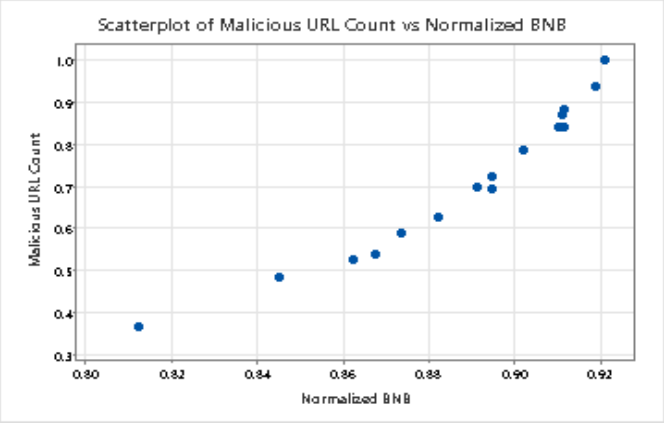


Figure 36: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized BNB

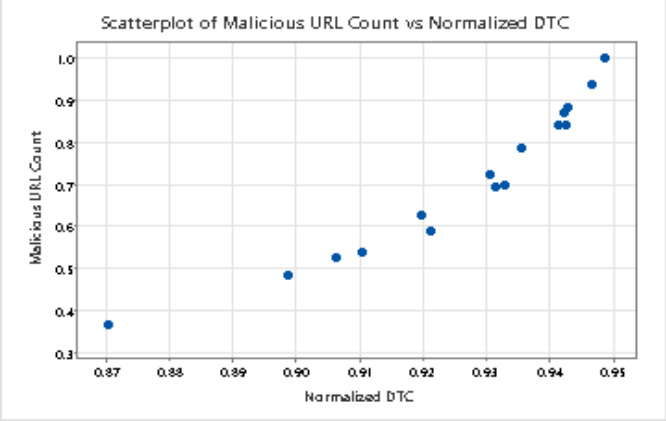


Figure 37: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized DTC

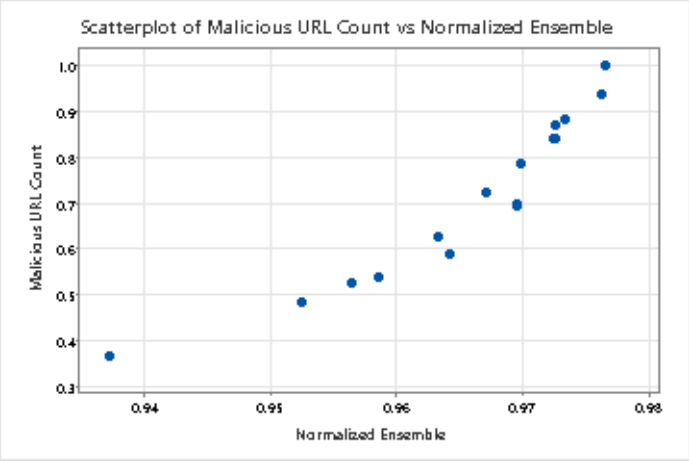


Figure 38: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized Ensemble

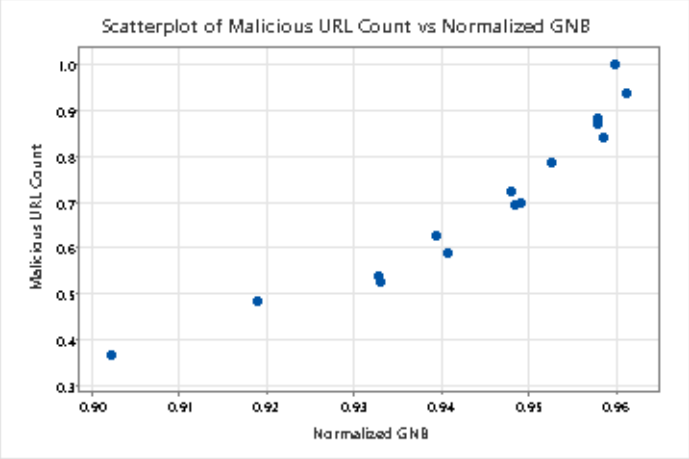


Figure 39: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized GNB

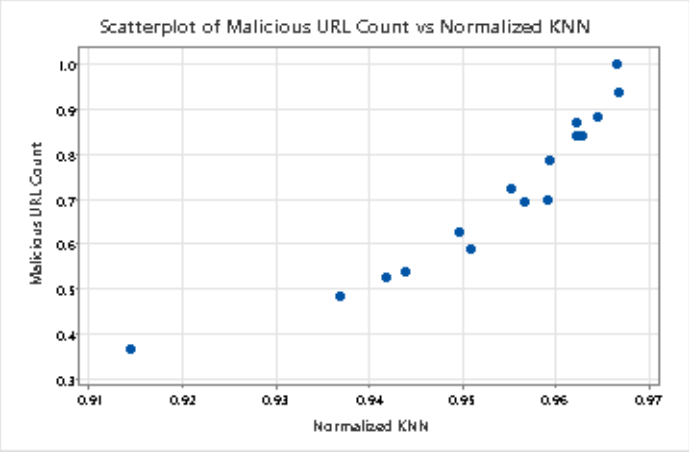


Figure 40: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized KNN

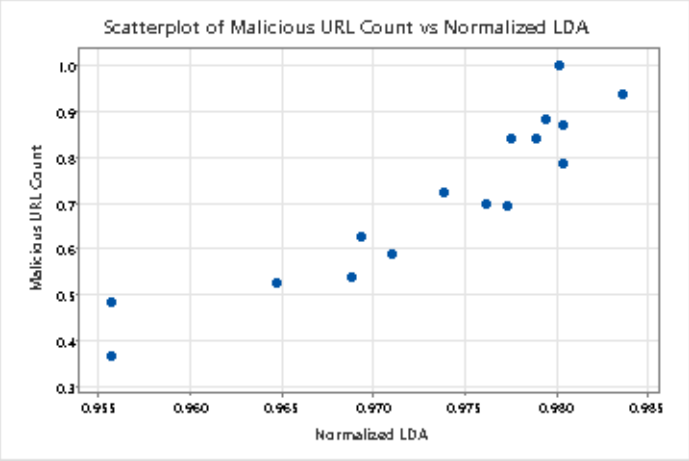


Figure 41: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized LDA

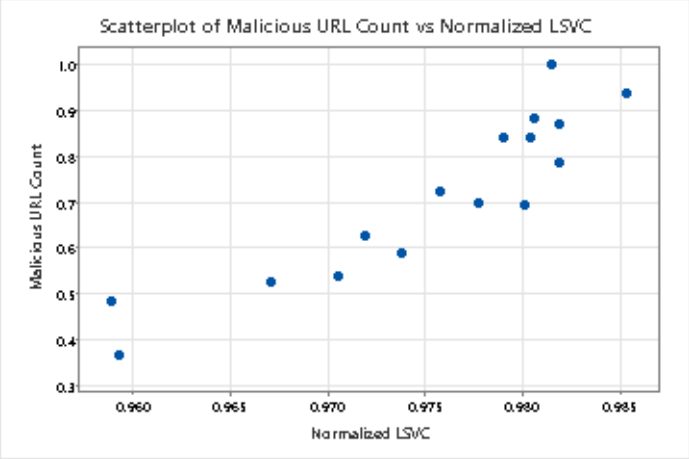


Figure 42: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized LSVC

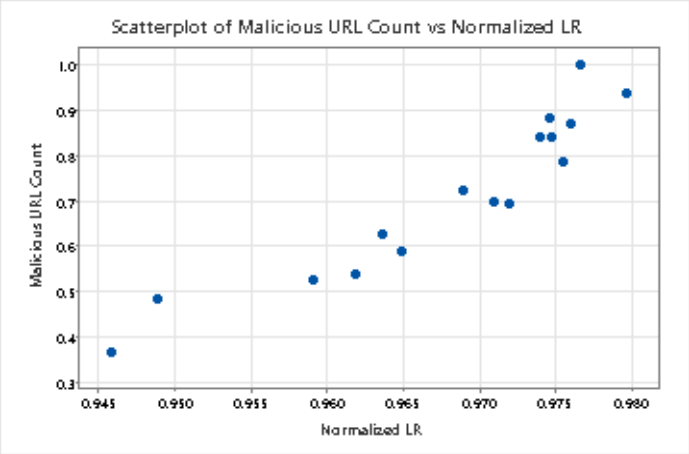


Figure 43: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized LR

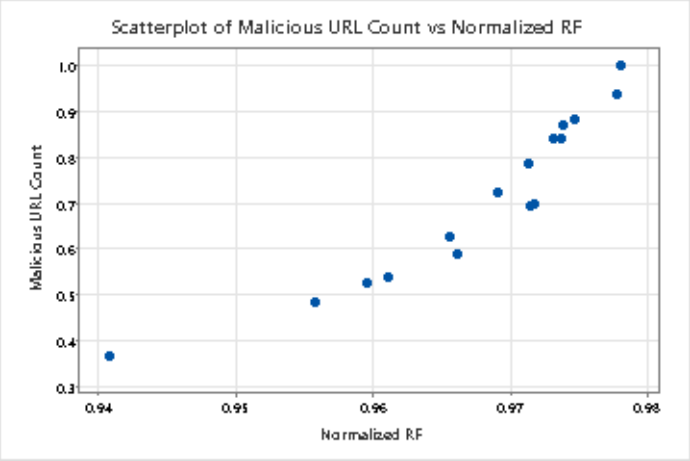


Figure 44: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized RF

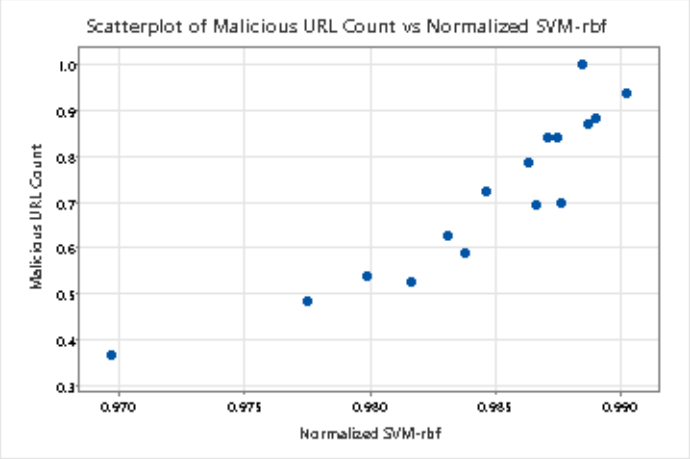


Figure 45: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized SVM-rbf

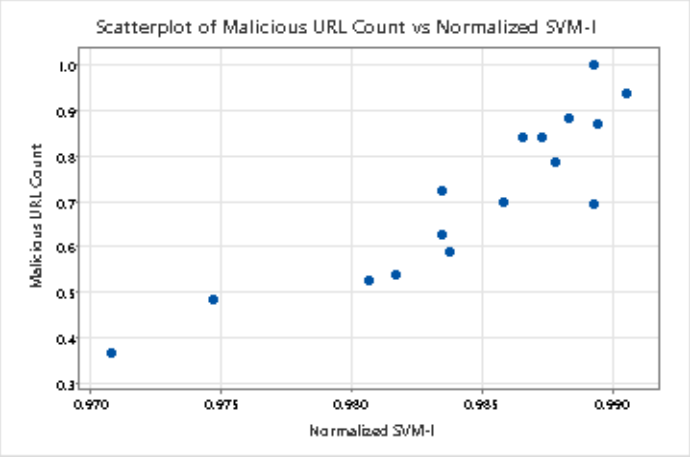


Figure 46: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized SVM-l

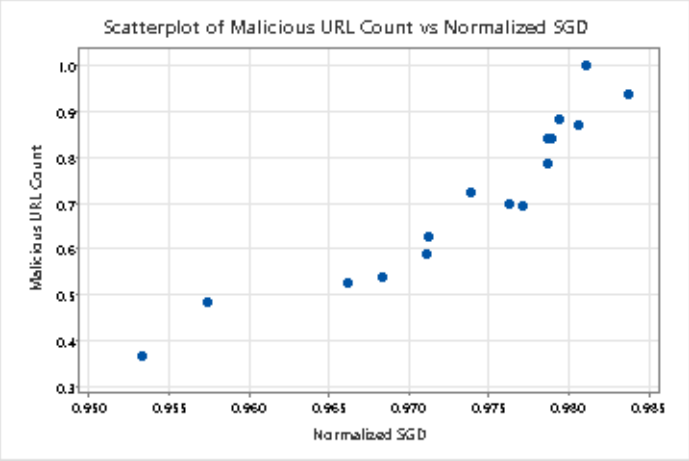


Figure 47: Scatterplot of Monotonic Relation between Malicious URL Count and Normalized SGD

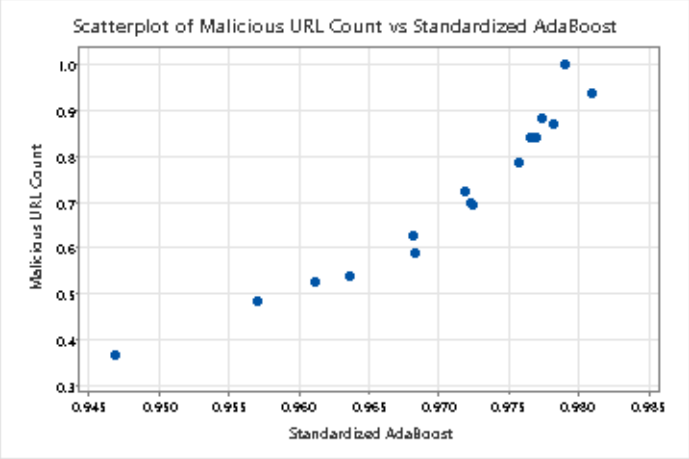


Figure 48: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized AdaBoost

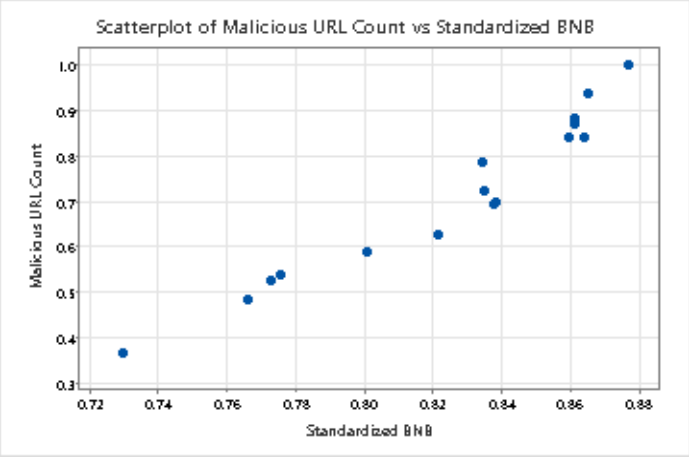


Figure 49: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized BNB

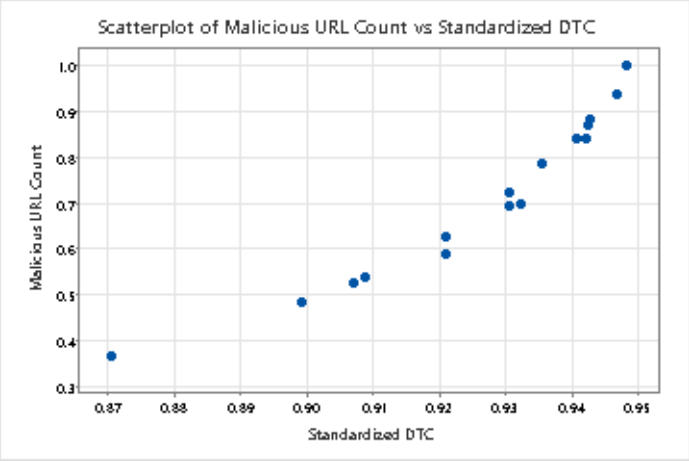


Figure 50: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized DTC

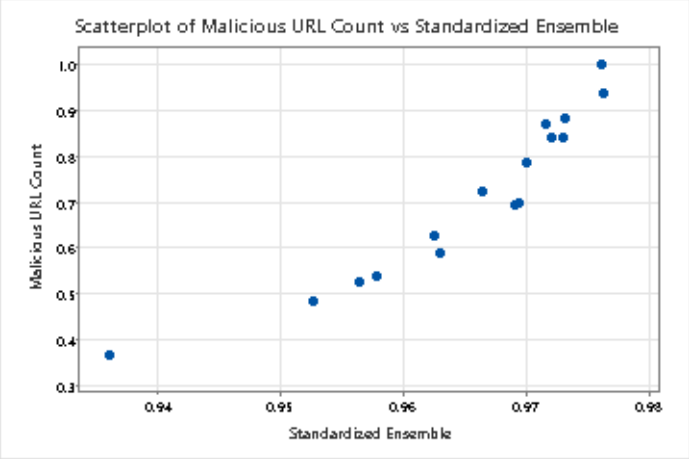


Figure 51: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized Ensemble

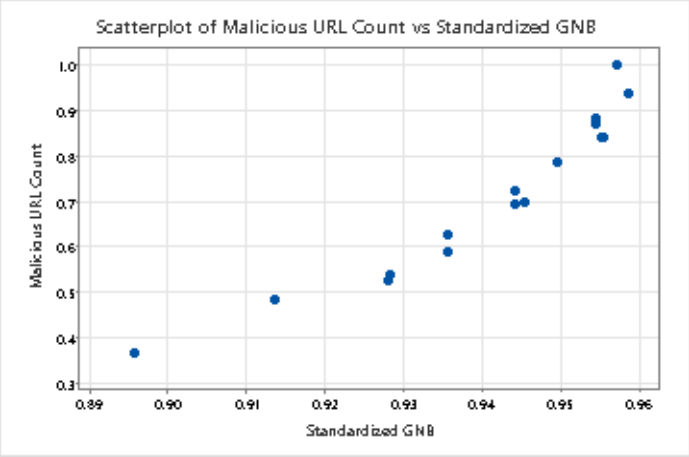


Figure 52: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized GNB

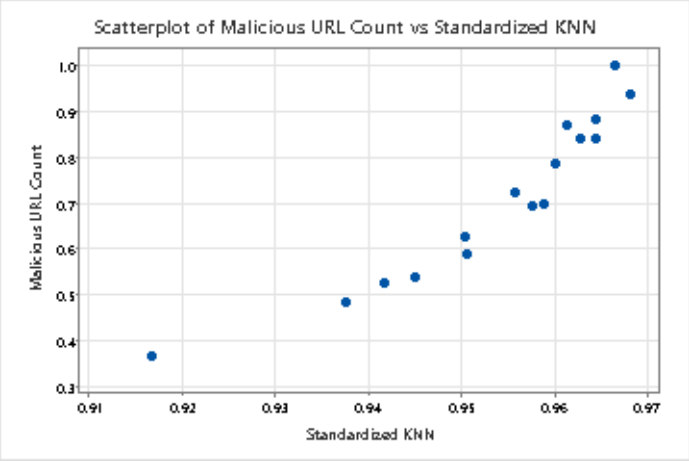


Figure 53: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized KNN

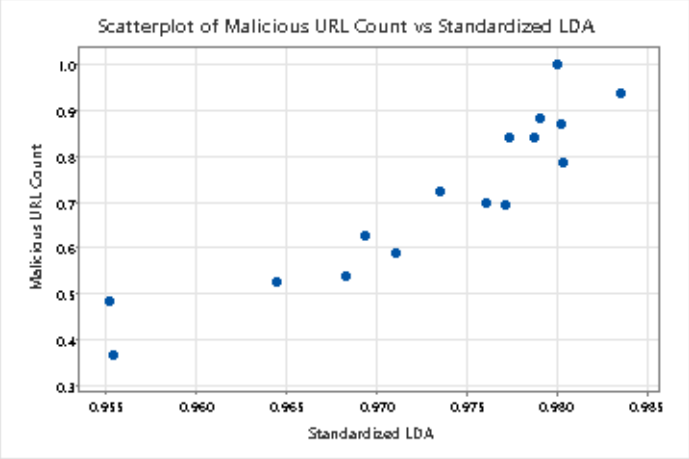


Figure 54: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized LDA

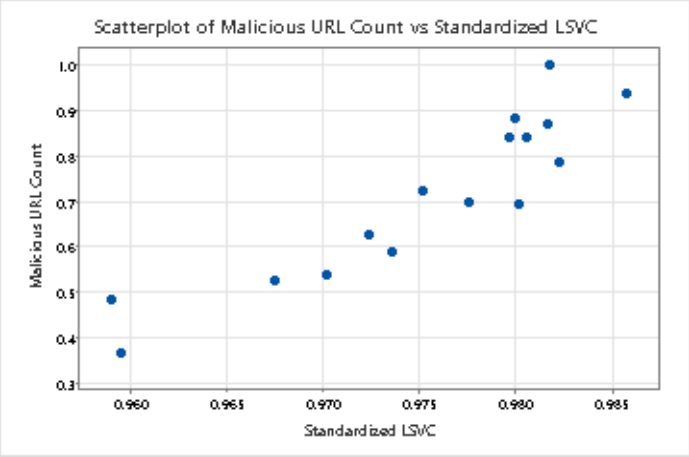


Figure 55: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized LSVC

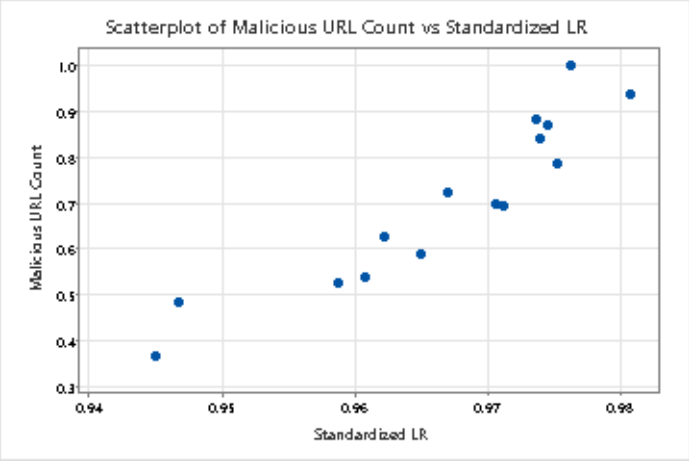


Figure 56: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized LR

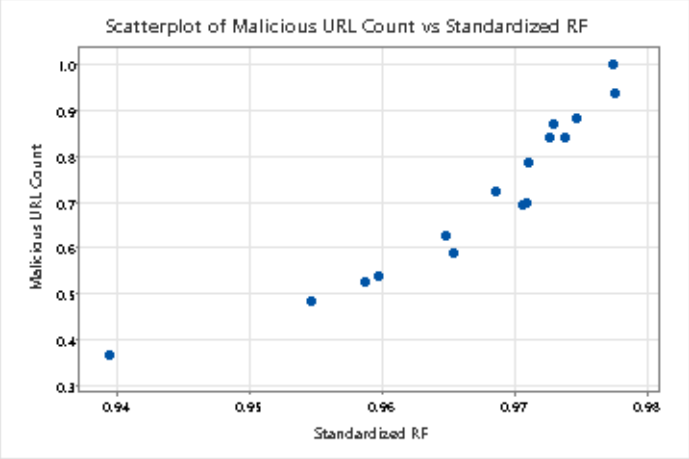


Figure 57: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized RF

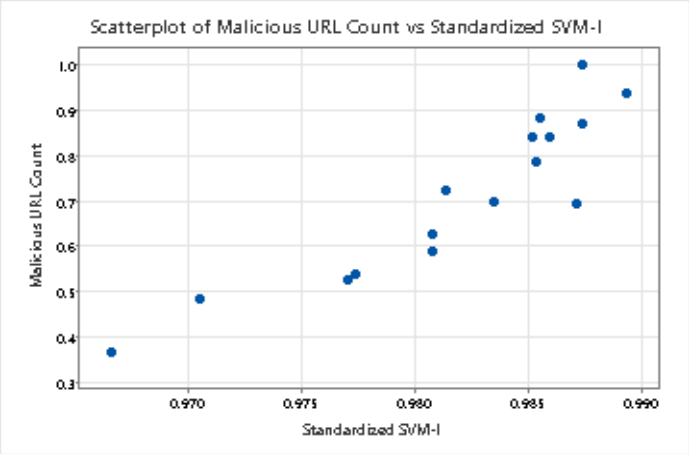


Figure 58: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized SVM-l

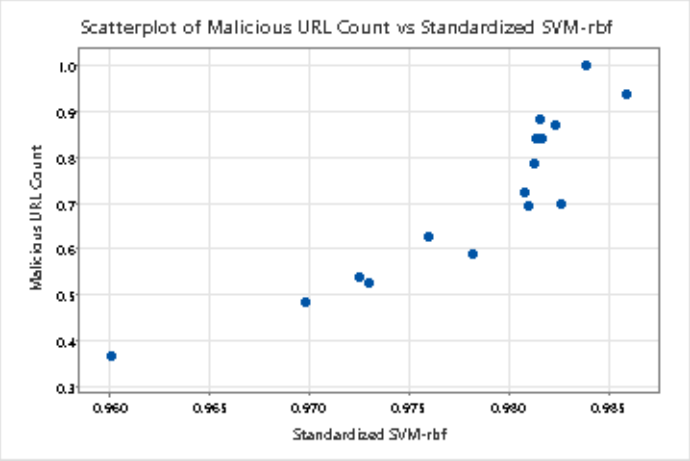


Figure 59: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized SVM-rbf

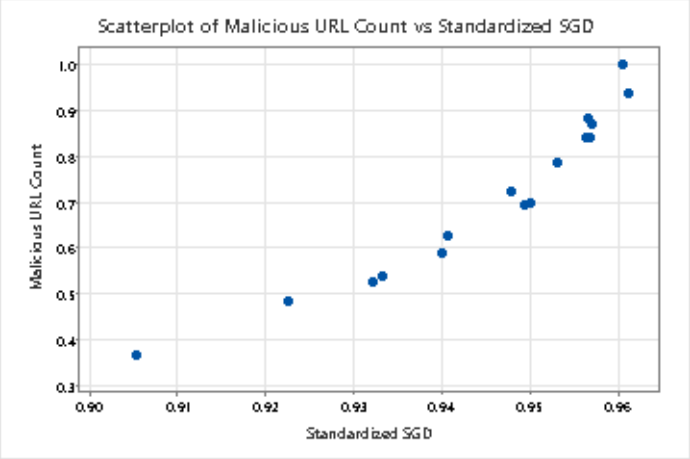


Figure 60: Scatterplot of Monotonic Relation between Malicious URL Count and Standardized SGD

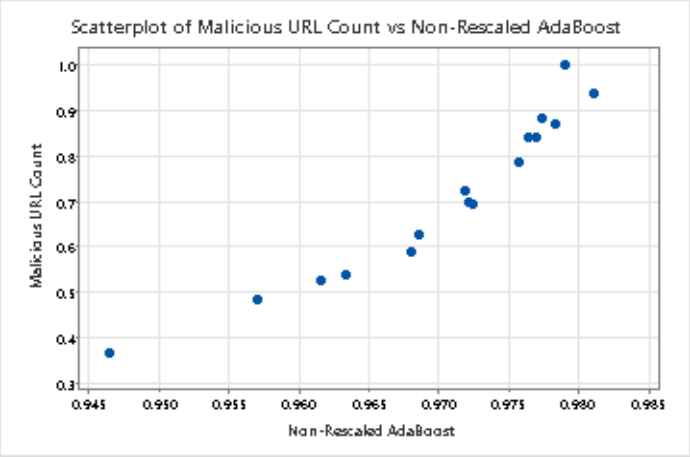


Figure 61: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled AdaBoost

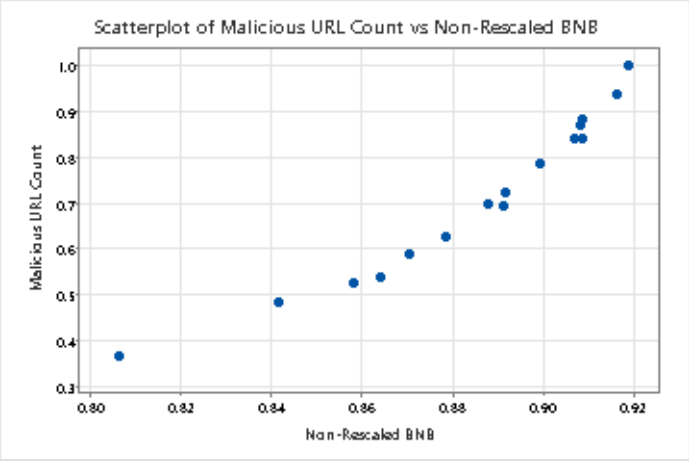


Figure 62: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled BNB

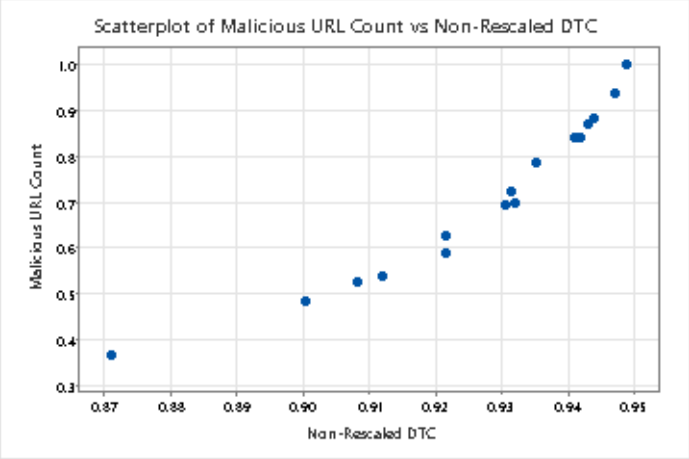


Figure 63: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled DTC

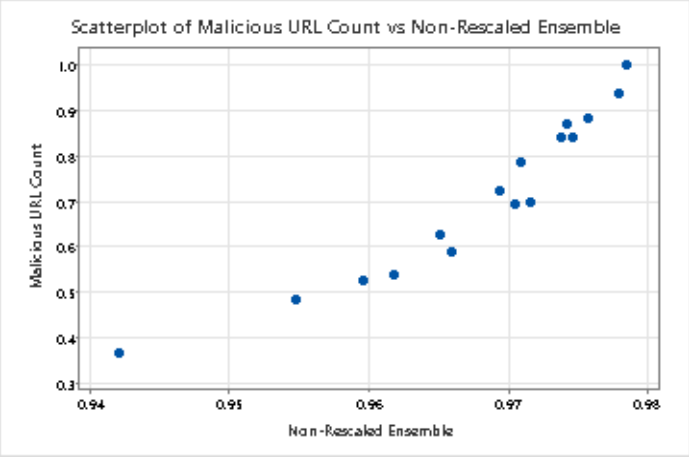


Figure 64: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled Ensemble

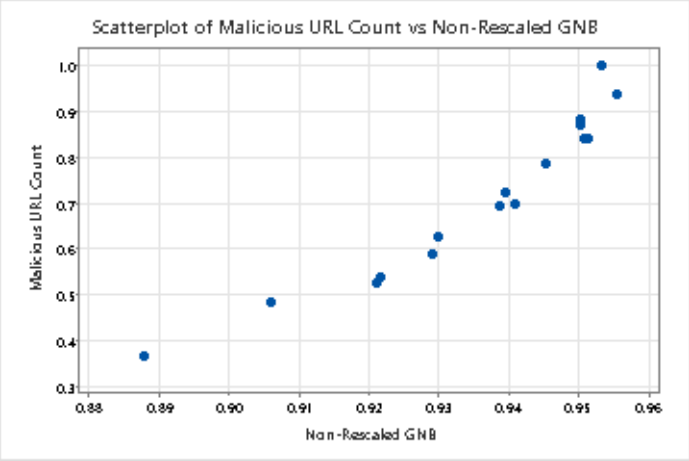


Figure 65: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled GNB

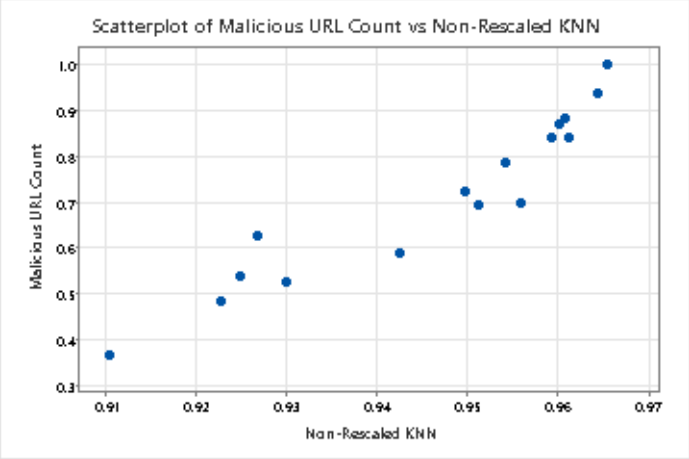


Figure 66: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled KNN

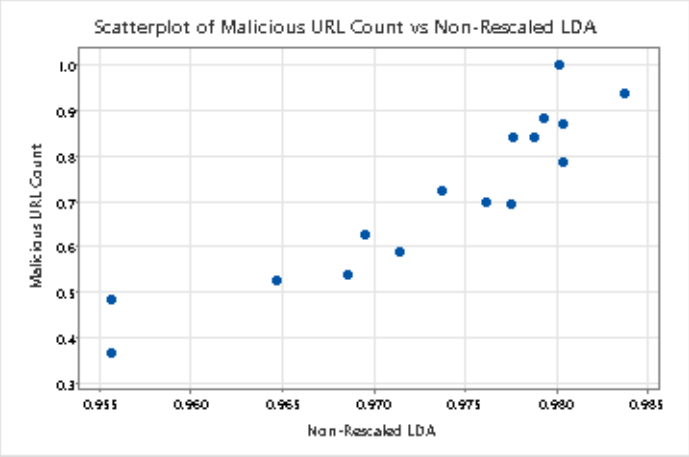


Figure 67: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled LDA

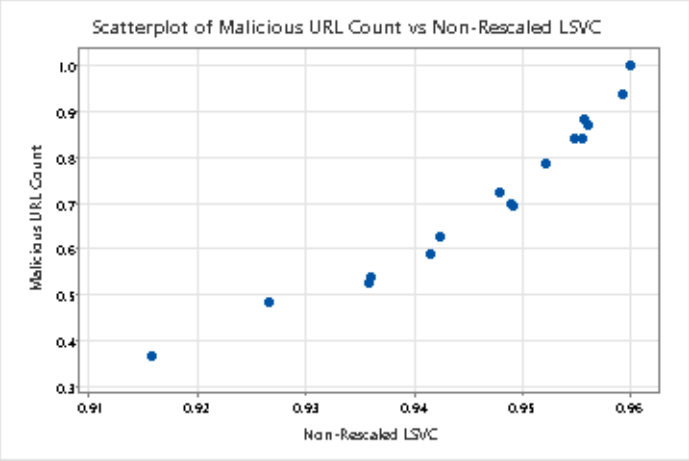


Figure 68: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled LSVC

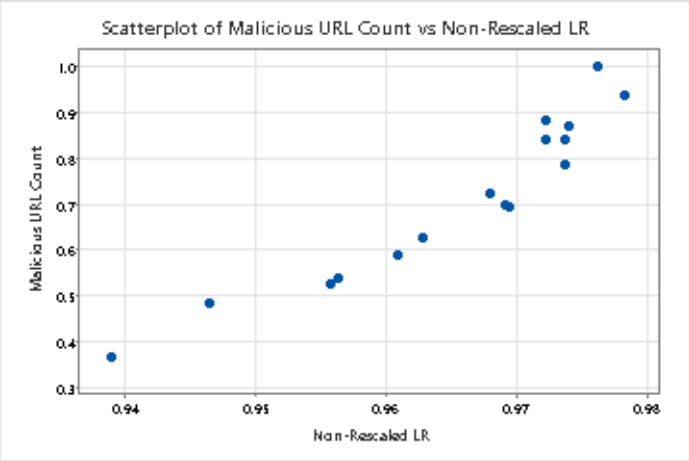


Figure 69: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled LR

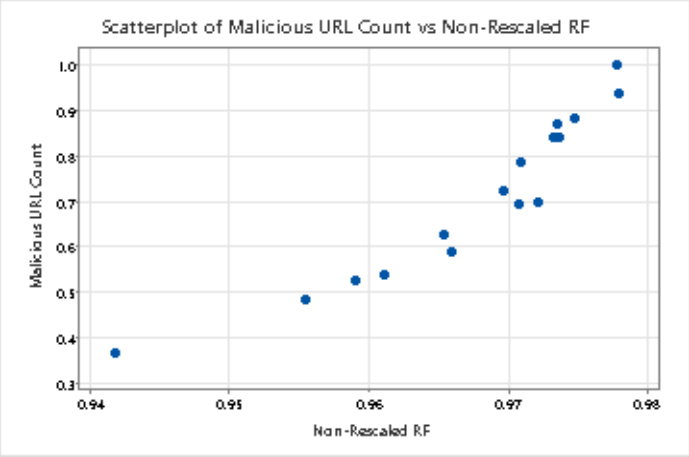


Figure 70: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled RF

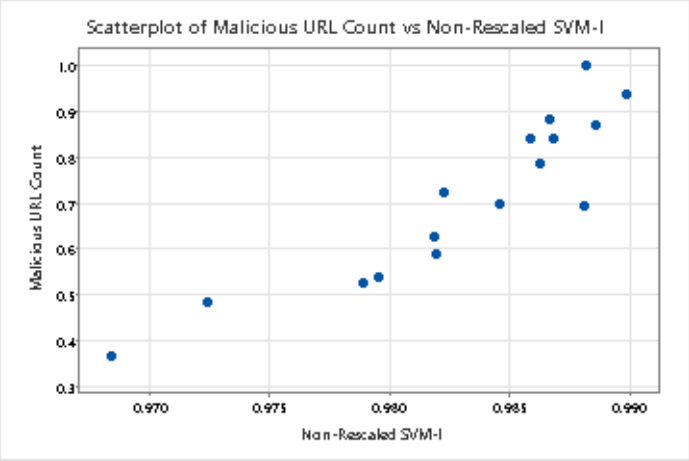


Figure 71: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled SVM-l

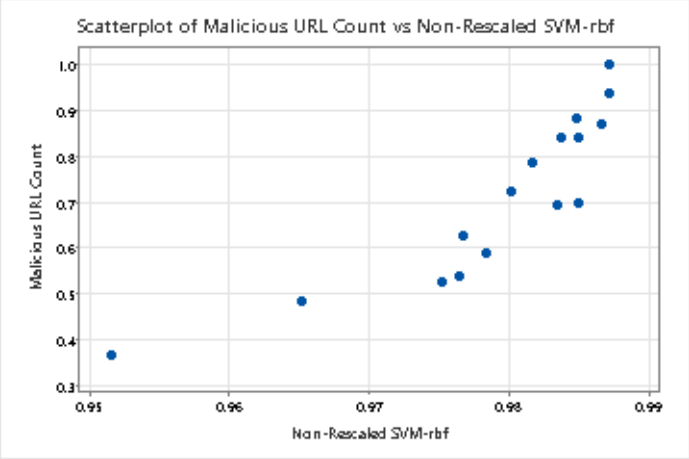


Figure 72: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled SVM-rbf

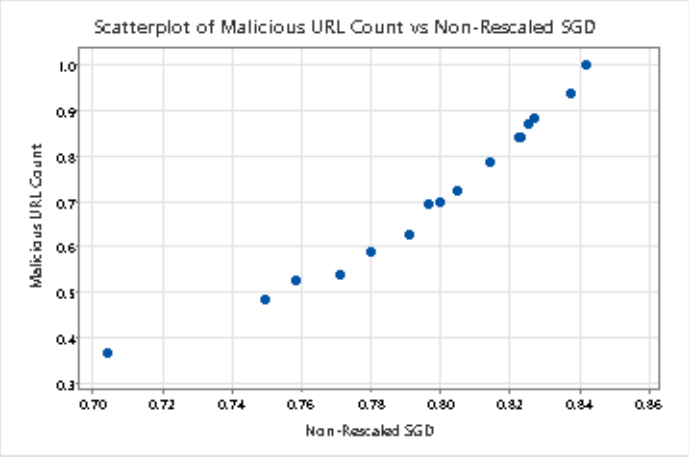


Figure 73: Scatterplot of Monotonic Relation between Malicious URL Count and Non-Rescaled SGD