UNIVERSITY
OF GÄVLE

FACULTY OF ENGINEERING AND SUSTAINABLE DEVELOPMENT
Department of Computer and Geospatial Sciences

Evaluation of machine learning models for classifying malicious URLs

Shayan Abad
Hassan Gholamy

2023

Degree project, Basic level (Professional degree), 15 HE
Computer Science
Study Programme in Computer Engineering

Supervisor: Anders Hermansson
Assistant supervisor: Mohammad Aslani
Examiner: Goran Milutinovic

Preface

We are pleased to present this bachelor thesis as part of our undergraduate studies.

The aim of this thesis was to investigate and evaluate different machine learning models for classifying malicious URLs. With the rapid growth of cybercrime and the increasing sophistication of cyberattacks, it has become crucial to develop robust methods for identifying and mitigating threats in online environments.

Throughout our research journey, we explored various machine learning models, including Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (KNNs), and Random Forest (RF). These models were trained and tested using carefully curated datasets comprising different types of URLs.

The thesis explores the evaluation of these models by employing instance selection methods, which are data reduction based on locality-sensitive hashing (DRLSH), boundary point extraction based on locality-sensitive hashing (BPLSH), and random selection. By implementing these techniques, we aimed to identify the most effective approach for selecting instances and optimizing the performance of the machine learning models.

It is our hope that this thesis contributes to the broader field of cybersecurity and aids in the development of effective techniques for combating the ever-evolving threats posed by malicious URLs. We would like to express our sincere gratitude to our supervisors, the experts at Sogeti and the IT-supporters at the University of Gävle for their support throughout this study.

Abstract

Millions of new websites are created daily, making it challenging to determine which ones are safe. Cybersecurity involves protecting companies and users from cyberattacks. Cybercriminals exploit various methods, including phishing attacks, to trick users into revealing sensitive information. In Australia alone, there were over 74,000 reported phishing attacks in 2022, resulting in a financial loss of over $24 million. Artificial intelligence (AI) and machine learning are effective tools in various domains, such as cancer detection, financial fraud detection, and chatbot development. Machine learning models, such as Random Forest and Support Vector Machines, are commonly used for classification tasks. With the rise of cybercrime, it is crucial to use machine learning to identify both known and new malicious URLs. The purpose of the study is to compare different instance selection methods and machine learning models for classifying malicious URLs.

In this study, a dataset containing approximately 650,000 URLs from Kaggle was used. The dataset consisted of four categories: phishing, defacement, malware, and benign URLs. Three datasets, each consisting of around 170,000 URLs, were generated using instance selection methods (DRLSH, BPLSH, and random selection) implemented in MATLAB. Machine learning models, including SVM, DT, KNNs, and RF, were employed. The study applied these instance selection methods to a dataset of malicious URLs, trained the machine learning models on the resulting datasets, and evaluated their performance using 16 features and one output feature.

In the process of hyperparameter tuning, the training dataset was used to train four models with different hyperparameter settings. Bayesian optimization was employed to find the best hyperparameters for each model. The classification process was then conducted, and the results were compared. The study found that the random instance selection method outperformed the other two methods, BPLSH and DRLSH, in terms of both accuracy and elapsed time for data selection. The lower accuracies achieved by the DRLSH and BPLSH methods may be attributed to the imbalanced dataset, which led to poor sample selection.

Keywords: Machine learning, Cyber security, Classification, Malicious URL, Instance selection

Table of contents

1 Introduction

Millions of new websites are created every day that collect user data through login functions. The large number of networks makes it difficult to determine which one is safe and reliable [1]. Cybersecurity can be defined as a set of tools or techniques aimed at protecting companies and users from cyberattacks [2, p. 2].

Cybercriminals use many ways to manipulate internet-users to give out sensitive and personal information and one of the most regular ways to do so is through malicious URLs which are hyperlinks. By interacting with these links, users expose themselves to a series of consequences, from the compromise of sensitive information to becoming a prime target for cyberattacks. The Australian Competition and Consumer Commission documented a total of 74.567 reported instances of phishing attacks in Australia during 2022, resulting in a cumulative financial loss exceeding 24 million of dollars [3].

1.1 Artificial intelligence

Artificial intelligence (AI) is the art of computing and is a machine that can perform specific tasks without the need for explicit programming. AI aims to mimic the human brain and reach its level of intelligence. AI has proven effective in many areas, including detecting and diagnosing cancerous tissues faster than doctors, detecting criminal schemes involving financial transactions, and developing chatbots that can speak and understand language and to become good helpers [4].

1.1.1 Machine learning

Machine learning is an AI application of systems, computers, or machine learning to identify patterns, structures, and categories from large amounts of data using statistical methods and algorithms. There are two main types of machine learning: supervised and unsupervised learning. Supervised learning is characterized by pre-defined data, providing access to both inputs and outputs of the model, whereas unsupervised learning does not use pre-defined datasets. There is also a subset of machine learning called deep learning, which mimics the human brain and has many layers [5, p. 429].

There are several well-known machine learning models that have been shown to be effective in classification, such as Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbors (KNNs), and Naive Bayes (NB) [6].

1.2 Problem description

With the widespread use of the internet, new forms of data theft have emerged, known as cybercrime. This involves breach of privacy through computer use and unauthorized access to confidential information and sensitive information. One of the most common methods is phishing, where the attacker tries to trick the recipient into revealing sensitive information, which can have serious consequences [7]. Defacement is another technique used by attackers to manipulate the content of web pages by changing the underlying code. This type of attack is used by hackers to exploit an organization's website [8, p. 142]. According to the Rivest-Shamir-Adleman (RSA) online fraud report, in 2014 almost 450,000 phishing websites lost $5.9 billion [9]. A blacklist with known URLs is used to search. However, this will not be particularly effective as new malicious URLs such as spam and phishing keep popping up. Machine learning models play an important role in identifying both known and new malicious techniques, such as phishing and spam. [10]. A Uniform Resource Locator (URL) is a way to get to a specific location on the Internet.

1.3 Purpose

The purpose of this study is to investigate and compare the performance of three different instance selection methods and four different machine learning models to classify malicious URLs.

The following research questions are answered in this study:

- What features play an important role in classifying malicious URLs in the dataset?
- Which machine learning models and instance selection methods lead to better performance on the dataset?

2 Theory

This section will discuss URLs, machine learning, classification, and hyperparameters, and review previous studies.

2.1 Uniform Resource Locators

A uniform resource locator (URL) is a unique address that identifies a resource, such as an HTML page. The URL consists of several parts, as shown in Figure 1. First, it is planned to specify the protocol that will be used to retrieve the object, with HTTPS (encrypted connection) and HTTP (unencrypted) being the most used protocols. The IP address indicates the web server being requested, and the port indicates the gateway that should be used to access content. The default port number for the HTTP protocol is 80 and for HTTPS it is 443 to access content. The third part of the URL is the path to the object. The fourth section is a list of parameters that can be used to specify keys and values that allow other actions to be performed. Finally, there is an anchor that allows you to jump to a specific section of the web page. Parameters and anchors may sometimes be excluded from the URL [11].

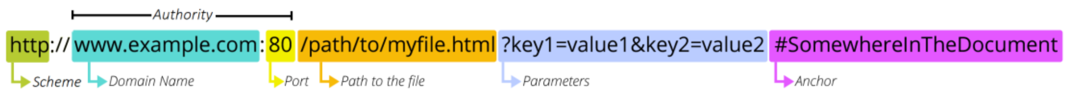

Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Figure 1. URL anatomy. Source: [11]

2.2 Machine learning

The most commonly used machine learning technique for classifying malicious URLs is classification, which is a kind of supervised machine learning, because the classification is based on prior information [12, p. 1310].

2.2.1 Classification

Classification is a supervised learning method for presenting and classifying data based on training data. This method involves training the algorithm with an already classified dataset, then using it to classify new data based on observed patterns in the training data [13]. The classification process is an important part of the task of classifying the data based on its features [14].

During the preparation phase, it is important for a machine learning classification model to obtain appropriate training, which involves carefully selecting a suitable dataset, cleaning it from null values and extracting relevant features. During the learning phase, a model is developed that can identify categories based on data and its features. In the evaluation phase, the model is tested with a test dataset that doesn't include samples used during the training phase. This allows it to evaluate the performance of the model with new data. [14].

2.2.2 Hyperparameters

Hyperparameters are structures that affect the training set of machine learning models. By adjusting these settings, the developer can improve the performance of the model, potentially leading to higher accuracy and shorter learning times [15]. Hyperparameter optimization is a process that attempts to optimize the hyperparameters of a model to achieve the highest performance on a dataset. As shown in Figure 2, machine learning uses training, validation, and testing processes to evaluate the performance of a model to avoid overfitting, which refers to a situation where the model excessively emphasizes each individual data point in the training dataset, resulting in the loss of its ability to generalize the data. Therefore, the model performs poorly when applied to new data. Various techniques can be employed to address this issue and enhance the model's performance with new data. [16].

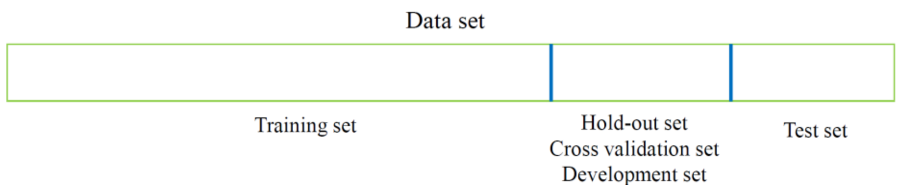


Figure 2. Training, validation, and testing processes. Source [1]

The training data is used to train the model. The validation data is a separate dataset that was not used in model training. The purpose of the validation data is to evaluate the performance of the model. After training and validation, testing of the data is used to evaluate how well the model performs on new datasets [17].

[1] M. Aslani," Introduction to Neural Networks", Presentation, University of Gävle, 6 March 2023.

Bayesian optimization is a method used to find the best settings for a model. It uses a surrogate model which is a prototype based on a set of points which represent different combinations of hyperparameters for the model. Then, it selects the next point to evaluate the objective function, f(x). The function uses the prototype model to determine the next optimization step. By doing this, it makes the process faster and requires fewer attempts. The most common ways to measure improvement are "likely to improve", "expected improvement", and "lower confidence limit". The most popular type of model used is called a Gaussian process model [18, pp. 30505, 30514].

The Gaussian process (GP) is a probabilistic regression model that helps us understand and predict unknown functions, denoted as f(x). It accomplishes this by utilizing a mean function, m(x), and a kernel function, k(x, x'). The mean function, m(x), provides an initial estimate or baseline for what the unknown function, f(x), might look like. It gives us a starting point to make predictions and understand the general behavior of the data. On the other hand, the kernel function, k(x, x'), allows us to capture the similarity or correlation between different data points. It quantifies the likelihood of two data points, x and x', having similar values. By measuring these relationships, we gain insights into the connections between data points, which in turn enables us to make predictions based on this information. [18, p. 30514].

2.3 Previous research

Previous research has been conducted on malicious URL detection and classification using various machine learning models, resulting in different percentages of accuracy. These studies have contributed to the understanding of effective approaches and techniques for identifying and classifying malicious URLs.

M.Sc. Aljabri [19] conducted a comprehensive analysis using over one million URLs with various attributes including IP address, geographic location, and HTTPS status. The dataset was divided into two parts: one for training with 1.2 million datasets and the other for testing with 0.364 million datasets. Twelve features were used in the study. The process was four steps, the dataset was analyzed to understand its existing features. Useful features were then extracted from the dataset. Alternative machine learning models, including NB and RF, were used to prepare inputs for machine learning models and to analyze the results, and the NB model was shown as the most accurate, with an accuracy of 95%.

In a previous study by Y. Li [20], a dataset of approximately 52,000 URLs was used, where 70% of the dataset was used for training and the remaining for testing. Eight features, such as HTTPS status, IP address, number of dots in the domain name, and top-level domain-related features were used in the work. The models used in the study were SVM, KNNs, DT, RF, gradient boosting decision tree, XGBoost (XGB), and LightGBM (LGB). The result showed that the SVM model had an accuracy of 94.45%.

An earlier study by A. Saleem Raja [21] used a dataset of about 66,000 URLs. In these cases, 70% was devoted to model training and 30% to testing. The models trained and tested included SVM, LR, KNNs, NB, and RF. About 20 features were used in the analysis, such as URL length, HTTPS status, number of digits, alphabetic characters, and symbols in URLs. RF showed the highest accuracy, reaching 99%, and SVM reached 98%.

In a previous study, S.H. Ahmed [7], they used a dataset of 3000 URLs, where 1500 of them were malicious and 1500 were benign. The objective of the study was to develop a machine learning model using RF, DT, LGB, LR, and SVM methods for malicious URL detection. 80% of the total data was used for training and 20% for testing. The model was trained with fifteen features including domain name, URL length, and HTTPS status. The results showed that LGB had the highest accuracy during training, with 89.5%, and 86% during testing. RF ranked second with 88.3% in training and 85.3% in testing.

The study conducted by [22] and published by Kaggle, a data analysis platform, collected a large amount of information from different categories, including 428,103 benign URLs that pose no risk to the user, 96,457 defacement URLs, 94,111 phishing URLs, along with 32,520 pieces of malware. The following models were used in this work: DT, RF, KNNs, new trees, Gaussian NB, stochastic gradient descent, and Ada-boost. These models were trained on different datasets in order to identify different malicious URLs and measure their accuracy. The three models, DT, RF, and trivial trees had the highest accuracy, at 91%.

A study has been conducted by A. Patil [23] using machine learning to detect malicious URLs. A total of 651,191 URLs were used and categorized as malware, defacement, benign, and phishing. The following models were used to detect and classify malicious URLs: XGB, LGB, and RF. To facilitate better decision-making by the model, feature extraction was carried out, such as the use of IP addresses, calculation of letters, digits, and non-alphanumeric characters. The result showed that RF had the highest accuracy of over 91%.

3 Process and results

This part of the paper presents the methods used and explains the results. The methodology section describes the classification process, which involves classifying the data based on its features. The results section includes a summary of the results obtained when applying the method.

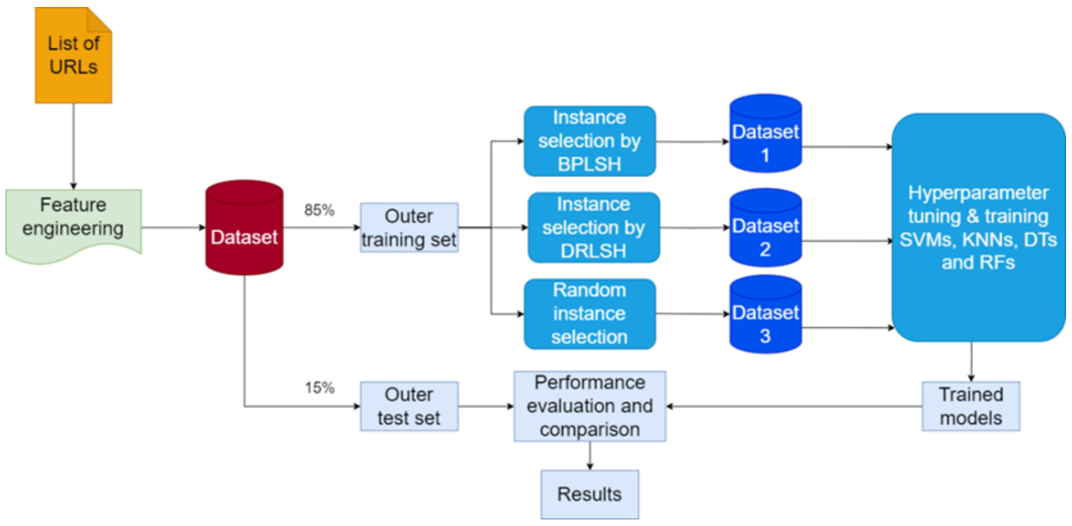
3.1 Methodology

The workflow consisted of three main phases (Figure 3). The first phase was data preparation, where relevant data was selected and cleaned for analysis. It was crucial to provide suitable data with high-quality features to train the model, based on previous research. Additionally, it was important to eliminate null values from the dataset as they could have a negative impact on the model's training during the training phase. When the dataset was checked, we did not find any null values. This means that there were no missing categories for URLs or any other missing information in the dataset. After this step, certain features that were utilized by previous studies to identify malicious URLs were selected and incorporated into the dataset. In this study, 85% of the original dataset was allocated for training purposes, while the remaining 15% was reserved for testing.

The machine learning models were trained on the training dataset to perform hyperparameter tuning and find the best point hyperparameters for the models using Bayesian optimization. These hyperparameters were then used on the reduced datasets in the training phase later. Following this, three datasets were generated using three different instance selection methods to reduce the number of samples, which led to a faster training and testing process.

The second phase, which was data learning, dealt with the development of models that could identify categories based on data and their features. The final phase was data evaluation, where the model was tested with new data not used in training.

To reduce the risk of overfitting the model, it was important to use K-fold cross-validation. Cross-validation involves splitting the dataset into multiple subsets or folds. In this study, five iterations of cross-validation were conducted. The MATLAB software, which was used in this study, chose five as the default value that could fit the dataset and its number of samples. Each iteration divided the dataset into five partitions. For example, in the first iteration of cross-validation, Folds 2, 3, 4, and 5 were combined and used as the training set, while Fold 1 was used as the testing set. This process continued until the other folds were completed. The decision to use five iterations was to obtain a robust assessment of the model's generalization capability across different subsets of data.



List of URLs
Feature engineering
Dataset
85%
Outer training set
15%
Outer test set
Instance selection by BPLSH
Instance selection by DRLSH
Random instance selection
Dataset 1
Dataset 2
Dataset 3
Hyperparameter tuning & training SVMs, KNNs, DTs and RFs
Trained models
Performance evaluation and comparison
Results

Figure 3. Workflow of the study.

3.1.1 Data collection

An existing dataset from data analysis platform Kaggle [25], with about 650,000 URLs representing URLs was used. This dataset consisted of four categories: phishing, defacement, malware, and benign URLs. It comprised two columns, one for the URL link itself and another for the URL type. However, it was observed that the dataset suffered from class imbalance, with 66% of the data belonging to benign URLs and the remaining portion distributed among the other categories. It was decided to train and test the models and methods using the imbalanced dataset in order to evaluate their accuracy and performance. This choice was made to assess how well the models and instance selection methods handle the challenges posed by imbalanced data.

Three datasets were generated from the total dataset. Each dataset consisted of around 170,000 URLs. Three instance selection methods were used in MATLAB to generate the datasets.

3.1.1.1 Random

A dataset consisting of random data was created by using the randperm function in MATLAB. The function was used to select around 170,000 random URLs from the training dataset. This instance selection method was the fastest method for reducing the data, taking only a few seconds.

3.1.1.2 BPLSH

The BPLSH data selection method, which has been developed by M. Aslani and S. Seipel [26], is utilized to select and retain instances that are in close proximity to the boundaries of two categories while eliminating non-essential instances that are distant from these boundaries, as can be seen in Figure 4. In the study's dataset, which consists of four categories namely benign, defacement, phishing, and malware, the BPLSH method examines the boundaries of these categories. This instance selection method was the slowest method for reducing the data, taking more than 1.5 hours.

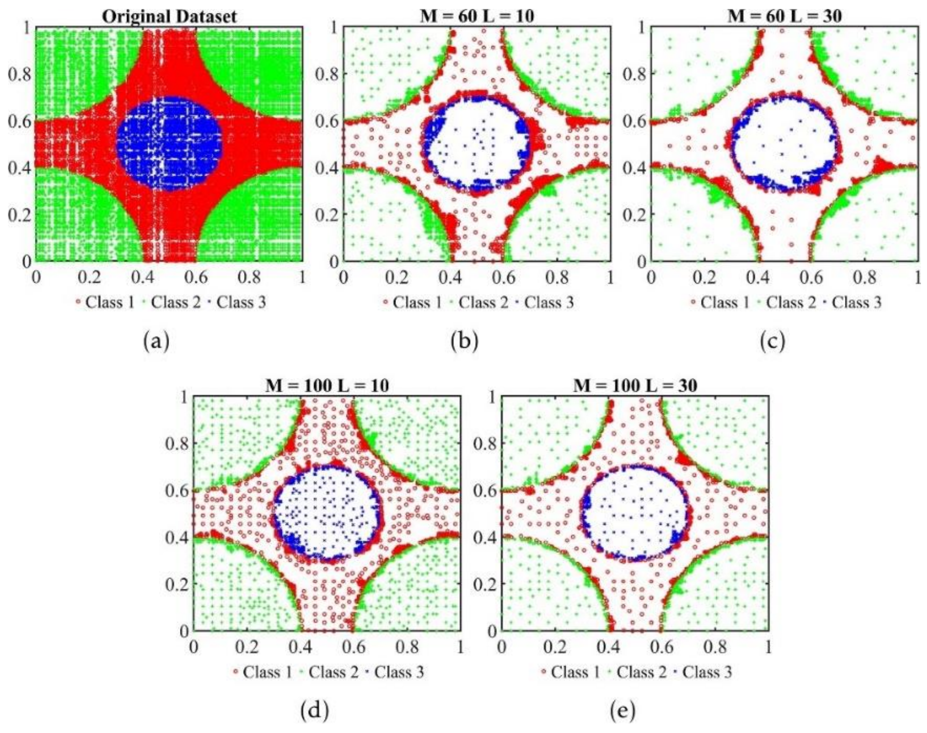


Figure 4. BPLSH process. Source [26]

3.1.1.3 DRLSH

Another data selection method, known as DRLSH, developed by M. Aslani and S. Seipel [27], is used to identify and eliminate similar samples. This method is faster than BPLSH and aims to reduce the amount of data shown in Figure 5, particularly in cases where two categories have similar samples. This can potentially result in a faster training speed. In our dataset, which consists of four categories namely benign, defacement, phishing, and malware, the DRLSH method examines similar samples. This instance selection method was faster than the BPLSH method for reducing the data, taking around 10-15 minutes.

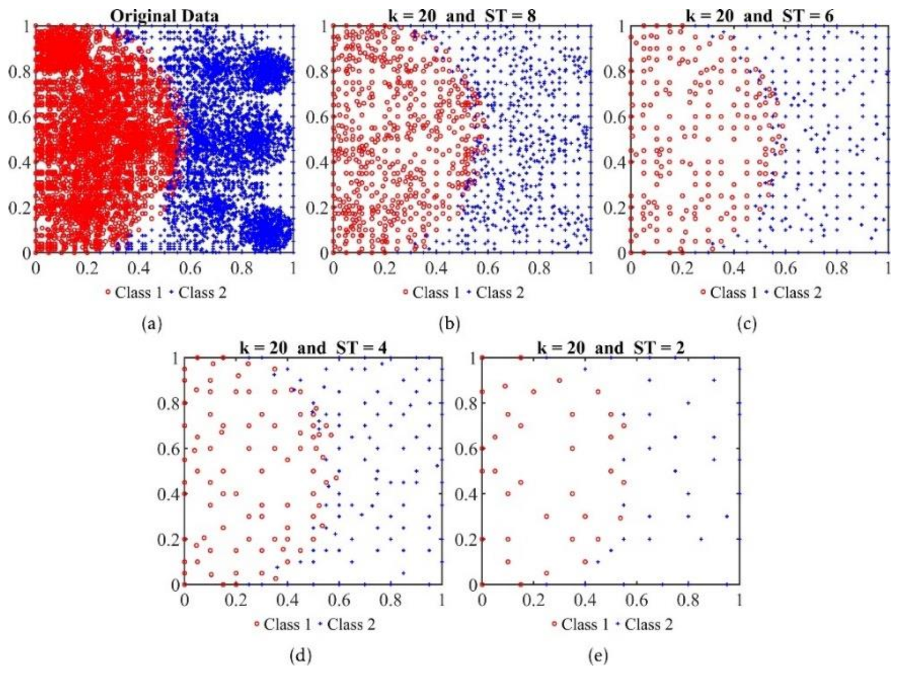


Figure 5. DRLSH process. Source: [27]

3.1.2 Feature engineering

In this study, 16 different features were utilized for training and learning machine learning models, along with one output feature (Table 1).

Table 1. Features list.

Feature	Data type	Description
Type (output feature)	Categorical	Categorization of URL type is based on whether it is phishing, defacement, malware, or benign.
Url_length	Numeric	The length of a given URL.
Domain_length	Numeric	The length of the domain name.
Has_ipv4	Numeric	Verifying the presence of an IPv4 address in each URL and subsequently returning either 1 or 0.

Has_http	Numeric	Verifying the presence of the HTTP protocol in each URL and subsequently returning either 1 or 0.
Has_https	Numeric	Verifying the presence of the HTTPS protocol in each URL and subsequently returning either 1 or 0.
Count_dots	Numeric	Calculation of the number of dots.
Count_dashes	Numeric	Calculation of the number of dashes.
Count_underscores	Numeric	Calculation of the number of underscores.
Count_slashes	Numeric	Calculation of the number of slashes.
Count_ques	Numeric	Calculation of the number of question marks.
Count_non_alphanumeric	Numeric	Calculation of non-alphanumeric characters.
Count_digits	Numeric	Calculation of the number of digits.
Count_letters	Numeric	Calculation of the number of letters.
Count_params	Numeric	Calculation of the number of parameters.
Has_php	Numeric	Verifying the presence of the word "php" in each URL and subsequently returning either 1 or 0.

Has_html	Numeric	Verifying the presence of the word "html" in each URL and subsequently returning either 1 or 0.

3.1.3 Models

Four models, namely Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNNs), and Support Vector Machine (SVM), were selected for comparison in this study. The choice of models was based on their popularity and prior usage in the classification area. DT create a tree-like structure to make decisions based on feature values. RF combine multiple decision trees for improved accuracy. KNNs assign class labels based on most of the k nearest neighbors. SVM create a hyperplane to separate different classes in high-dimensional space.

3.1.3.1 Decision Tree

This model is commonly used in data science and machine learning to classify data based on its features. This model generates a tree structure consisting of decision nodes representing different paths in the decision process. This approach makes it easier to identify the most important decision factors and how they affect the final outcome [28, p. 2094].

Bayesian optimization was utilized to optimize the decision tree model by identifying the optimal combination of hyperparameters that maximize the model's accuracy. The maximum number of decision nodes in the tree was set to 1 to reduce the number of decision nodes and the complexity of the tree. The split criterion was employed to measure the quality of the various divisions in the tree.

$$Gini = 1 - \sum_{i=1}^{n} (\rho_i)^2$$

Gini's diversity index is a method for measuring segregation parameters in decision trees models, where ρ_i represents the proportion of the total number of group members belonging to a particular class. In this case, it was used to assess how the tree correctly divides the data into different classes.

3.1.3.2 K-Nearest Neighbors

K-nearest neighbors (KNNs) is a machine learning model for classifying data based on similarity between observations. It works by checking the k-nearest neighbors of a given object in the training data and classifies the new samples based on their classification [29, p. 19].

To optimize the k-nearest neighbors' model, Bayesian optimization was employed. The number of nearest neighbors is a hyperparameter that affects the number of neighbors that the model considers when making predictions, and this value was set to 1. The distance weight was set to equal, indicating that all nearest neighbors have the same weight when utilized to make a prediction.

$$d(x,y) = \sqrt{((x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x\_n - y\_n)^2)}$$

The distance metric utilized to calculate the distance between two data points in a multidimensional space was set to Euclidean, where x and y represent two points in the space, (x_1, x_2, ..., x_n) and (y_1, y_2, ..., y_n) denote the coordinates of the n dimensions.

3.1.3.3 Random Forest

A well-known classification model is random forest, which is an ensemble learning model that generates numerous individual learners and combines their results. Random Forest selects the optimal parameter value at each node in a decision tree by randomly selecting a set of features. This random selection of features has its benefits. For instance, it assists the model in dealing with many features in each feature vector and reduces the internal dependency (correlation) between features. This makes the model less sensitive to internal disturbances and inaccuracies in the data [30, p. 664].

Bayesian optimization was used to optimize a random forest model, using the Bagging ensemble method which is a specific technique for building a random forest by randomly selecting a subset of trees and combining their results. The learner type was set to decision trees. The maximum number of splits, which is a hyperparameter that controls how many times the tree can be split into two nodes, was set to 20. The number of learners, which determines how many decision trees to build in the model, was set to 30.

3.1.3.4 Support Vector Machine

SVM is a common model in supervised learning techniques. The model is suitable for classification problems. The goal is to identify the optimal hyperplane that can best separate two categories in a dataset. The hyperplane is a line or plane that maximizes the distance between the closest points of the two categories [31].

To optimize the SVM model, Bayesian optimization was used with 30 iterations. A linear kernel was used to find a linear separating plane between the categories, and the one-vs-one multiclass method was used to classify each pair of categories.

3.2 Results

This section presents the results of the three datasets to evaluate the performance and accuracy of the models.

3.2.1 Hyperparameters tuning

In the hyperparameters tuning process, the training dataset was utilized to train those four models for each of its hyperparameter and try different methods to find the best settings för a models hyperparameters. The table for the best point hyperparameters shows the results for each model, which were obtained using Bayesian optimization (Table 2).

The default value for K-fold cross-validation in MATLAB was chosen, and it was set to five. The reason behind this choice was that 5-fold cross-validation is considered a standard and widely accepted practice [32]. It strikes a good balance between accurately estimating the performance of the model and managing the computational cost involved in the cross-validation process. The model was trained and evaluated five times, and the average value was then calculated to determine the best point hyperparameters.

Every model uses different settings for its hyperparameters. For example, DT has two hyperparameters, the first is maximum number of splits which refers to the maximum number of divisions that can be found in the tree. It reduces the complexity of the tree by determining the maximum number of nodes that can be created by splitting. The second hyperparameter is the split criterion, which specifies the method used to evaluate and select the best feature for splitting at each node in a tree.

The KNNs model's optimal hyperparameters were determined as follows: the number of neighbors was set to 10. This means that when making predictions, the model will consider the 10 nearest data points to a new data point and utilize their information. Additionally, the distance weight was set to "squared inverse." This choice of weight implies that the closer data points will have a stronger influence on the prediction.

RF using the following hyperparameters: ensemble method, which is used to reduce the overfitting and improve the model's generalization and robustness by building multiple decision trees and combining them which leads to a better classification. Number of learners which refers to the number of decision trees that should be created in the random forest model. Maximum number of splits, which refers to the maximum number of divisions that can be found in the tree, it reduces the complexity of the tree by determining the maximum number of nodes that can be created by splitting. Number of predictors to sample, which is used to randomly select several predictors (attributes) for every decision tree. The number specifying how many predictors are randomly selected at each split point in the tree.

The SVM model's optimal hyperparameters were determined as follows: multiclass method was "One-vs-One." This means that the SVM model creates multiple binary classifiers, comparing two classes at a time. It then combines the results of these classifiers to make predictions for multiple classes. Box constraint level, which is a regularization parameter in the SVM model that helps control the trade-off between achieving a low training error and maintaining a simpler decision boundary, was set to 943.1384. Kernel scale was set to 2.8684, which is a parameter that determines the reach or influence of each data point in the feature space. Kernel function specifies the type of transformation used to map the original data points into a higher-dimensional feature space. In this case, the chosen kernel function was the Gaussian kernel. Standardize data was set to true, which means that the input features were standardized before training the model.

Table 2. Information on the model's bestpoint hyperparameters as well as its tuning time.

Model	Best point hyperparameters	Tuning time
DT	Maximum number of splits: 36951	8 minutes
	Split criterion: Gini's diversity index	
KNNs	Number of neighbors: 10	21 hours
	Distance weight: Squared inverse	
RF	Ensemble method: Bag	2 hours
	Number of learners: 11	
	Maximum number of splits: 34067	
	Number of predictors to sample: 6	
SVM	Multiclass method: One-vs-One	46 hours
	Box constraint level: 943.1384	
	Kernel scale: 2.8684	
	Kernel function: Gaussian	
	Standardize data: true	

3.2.2 Test dataset

For the testing phase, a test dataset comprising approximately 97,500 URLs, which accounted for 15% of the original dataset, was utilized. Each category within the test dataset had varying numbers of observations. The dataset was used to test each model in each instance selection method. (Figure 6).

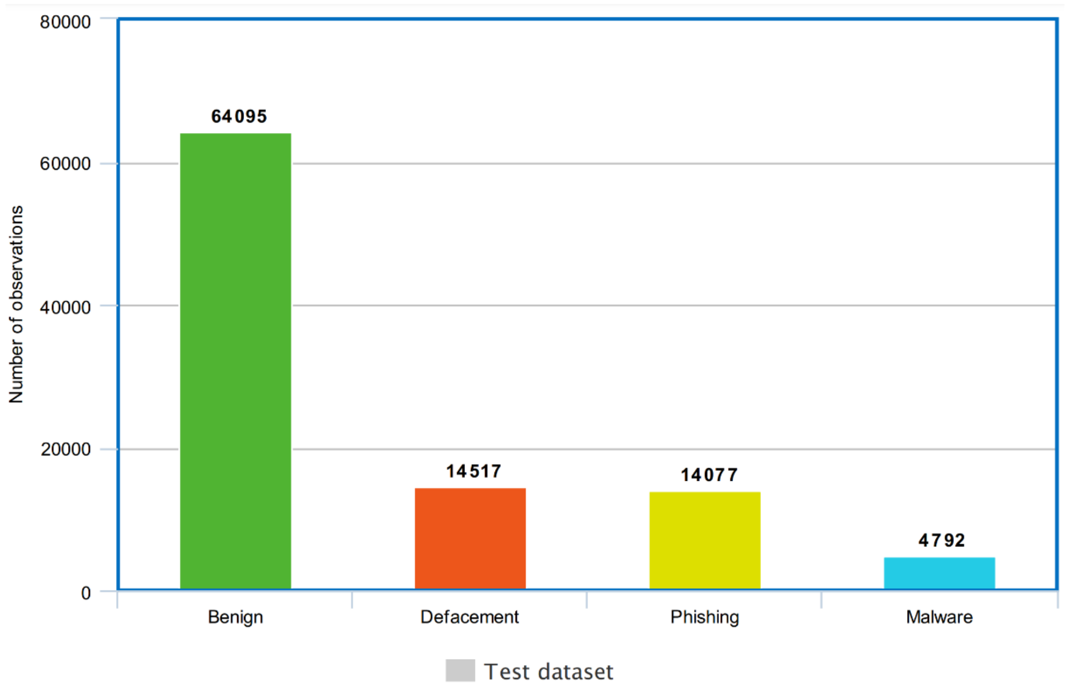


Figure 6. Number of observations for each category in the test dataset.

3.2.3 Dataset obtained by random

From this section onwards until the conclusion of the results, the datasets were generated using the instance selection methods and were then employed to train the models. When the random dataset was generated, it randomly selected around 170,000 samples from the training dataset. As a result, most of the selected samples belonged to benign URLs (Figure 7).

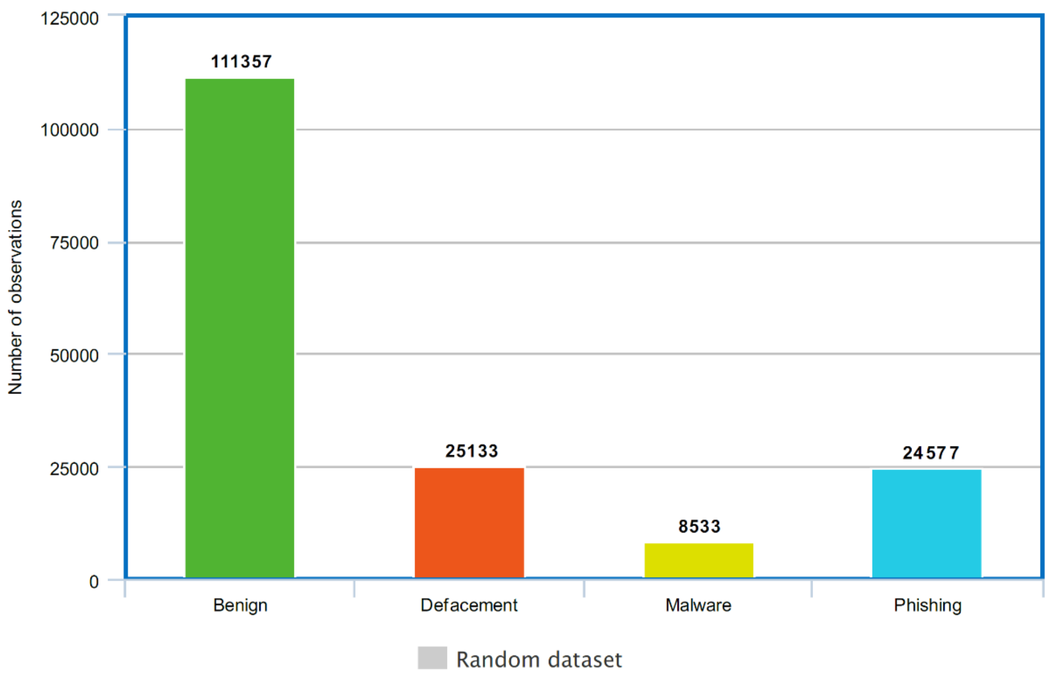


Figure 7. Number of observations for each category in the random dataset.

RF exhibits the highest accuracy compared to the other models (Table 3). Furthermore, considering the relatively short training time, this model can be considered as the best choice as an optimal model for this instance selection method.

Table 3. Information on the model's training and testing accuracy as well as its training time.

Model	Training accuracy	Test accuracy	Training time
RF	94.1%	94.4%	71 seconds
SVM	93.5%	93.8%	10 793 seconds
DT	93.0%	93.1%	16 seconds
KNNs	89.8%	90.2%	94 seconds

Test confusion matrix summarizes the performance of a classification model on a test dataset. The confusion matrix for the RF model shows that the accuracy rates for all categories, except for malware URLs, were more than 90% (Appendix A). Similarly, the DT model displayed higher accuracy rates across all categories, except for malware URLs, where it showed some degree of inaccuracy by incorrectly classifying certain malware URLs as phishing.

The confusion matrixes for the SVM and KNN models reveal that these models achieved high accuracies in predicting benign and defacement URLs (Appendix B). However, they demonstrated lower accuracies in the other two categories, namely malware and phishing.

3.2.4 Dataset obtained by DRLSH

When the DRLSH dataset was generated using its method, it selected around 170,000 samples that were not similar to each other from the four categories of URLs. As a result, most of the selected samples belonged to benign URLs (Figure 8).

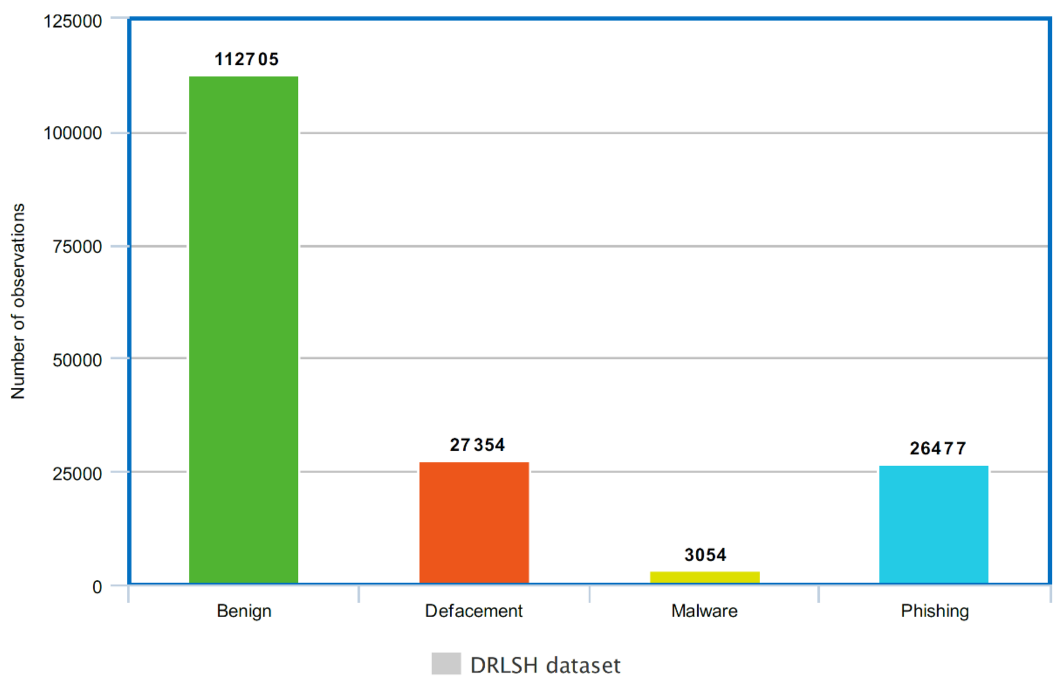


Figure 8. Number of observations for each category in the DRLSH dataset.

SVM model achieved higher accuracies in both the training and testing phases compared to the other models (Table 4).

Table 4. Information on the model's training and testing accuracy as well as its training time.

Model	Training accuracy	Test accuracy	Training time
SVM	90.9%	92.4%	18 390 seconds
RF	89.7%	90.1%	82 seconds
DT	87.7%	87.7%	21 seconds
KNNs	80.8%	87.4%	92 seconds

The test confusion matrixes for the RF and DT models display that these models achieved high accuracies in predicting benign and defacement URLs (Appendix C). However, they demonstrated lower accuracies in the other two categories, namely malware and phishing.

The confusion matrixes for both the SVM and KNNs models show that both models achieved higher accuracies in predicting benign and defacement URLs. However, the other two categories had lower accuracies (Appendix D).

3.2.5 Dataset obtained by BPLSH

When the BPLSH dataset was generated using its method, it selected around 170,000 samples that were closest to the boundaries of the four categories of URLs. As a result, most of the selected samples belonged to benign URLs (Figure 9).

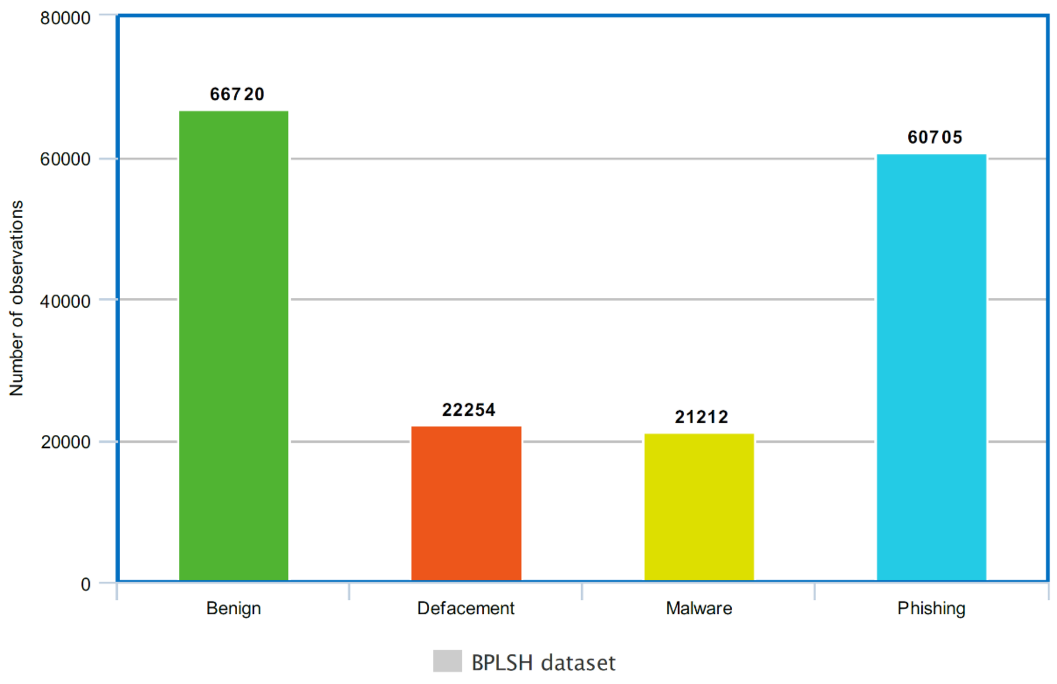


Figure 9. Number of observations for each category in the BPLSH dataset.

The RF model exhibited the highest test accuracy among all models (Table 5). This suggests that the RF model is better suited for handling new datasets than other models. However, the method's overall accuracy was lower than that of other instance selection methods.

Table 5. Information on the model's training and testing accuracy as well as its training time.

Model	Training accuracy	Test accuracy	Training time
RF	83.9%	88.1%	75 seconds
SVM	83.3%	86.5%	16 681 seconds
DT	82.6%	83.5%	23 seconds
KNNs	78.6%	71.1%	88 seconds

The confusion matrices for the RF and DT models indicate that the models accurately predicted defacement and malware URLs (Appendix E). However, the accuracies for the other categories were below 90%.

The confusion matrix for the SVM model reveals that it achieved an accuracy of approximately 92% in predicting defacement and malware URLs, while the accuracies for the other two categories were below 90% (Appendix F). On the other hand, the confusion matrix for the KNN model reveals that it performed better in predicting malware URLs compared to the other three categories, achieving an accuracy of 92.4%.

3.2.6 Feature importance

The feature "has_http" played the most important role in classifying malicious URLs (Figure 10). URLs that use HTTP as protocol are not encrypted and have a high probability of being a malicious URL. The calculation used to obtain the feature importance was Minimum Redundancy Maximum Relevance (MRMR). MRMR is a mathematical algorithm that leverages statistical measures to perform feature selection. The MRMR values represent the total MRMR score for each feature in all three datasets that were generated by instance selection methods.

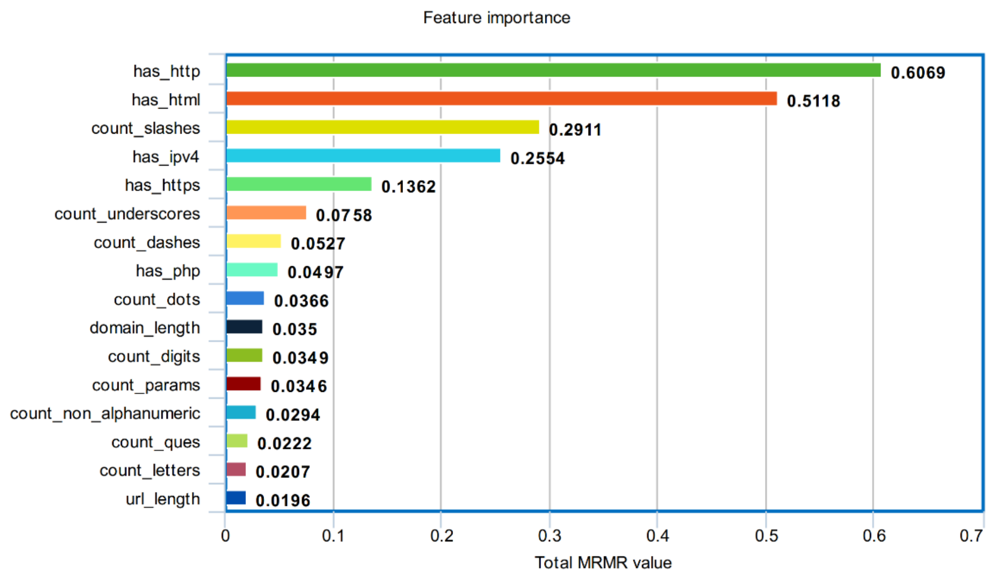


Figure 10. Feature importance.

4 Discussion

In this study, we have determined that the BPLSH and DRLSH instance selection methods are not suitable for reducing imbalanced datasets. This means that most of the selected samples belong to the category with the highest number of samples, leading to poor results for other categories with fewer samples.

Table 6. Information on the model's training and testing accuracy as well as its training time in random dataset.

Model	Training accuracy	Test accuracy	Training time
RF	94.1%	94.4%	71 seconds
SVM	93.5%	93.8%	10 793 seconds
DT	93.0%	93.1%	16 seconds
KNNs	89.8%	90.2%	94 seconds

Table 7. Information on the model's training and testing accuracy as well as its training time in DRLSH dataset.

Model	Training accuracy	Test accuracy	Training time
SVM	90.9%	92.4%	18 390 seconds
RF	89.7%	90.1%	82 seconds
DT	87.7%	87.7%	21 seconds
KNNs	80.8%	87.4%	92 seconds

Table 8. Information on the model's training and testing accuracy as well as its training time in BPLSH dataset.

Model	Training accuracy	Test accuracy	Training time
RF	83.9%	88.1%	75 seconds
SVM	83.3%	86.5%	16 681 seconds
DT	82.6%	83.5%	23 seconds
KNNs	78.6%	71.1%	88 seconds

Using the three different instance selection methods (random, DRLSH, and BPLSH) to reduce the dataset size has shown that achieving fair instance selection can be highly challenging, resulting in a significant impact on the results. This leads to poor classification performance, emphasizing the importance of starting with a more balanced dataset from the outset. Unfortunately, due to time limitations, we were unable to collect a sufficiently large dataset for all four categories of URLs. Therefore, there is potential for improving this study with more time and resources.

For example, performing hyperparameter tuning for SVM alone took approximately 46 hours, demanding continuous operation of our personal computer and substantial utilization of remote memory space. Throughout the training phase, we encountered several instances where the process was interrupted due to insufficient memory space on the remote desktop, necessitating us to restart the training procedure.

Initially, we had the expectation that the two instance selection methods, BPLSH and DRLSH, would yield higher accuracies compared to the random instance selection method. This was based on the understanding that BPLSH and DRLSH could select boundaries and eliminate similar samples, potentially improving the training process. However, contrary to our expectations, the random method resulted in the highest accuracies.

We observed that the models exhibited different accuracies across the datasets. However, SVM and RF achieved higher accuracies compared to the other two models, KNNs and DT. When considering the training and hyperparameter tuning times, RF emerges as a more favorable choice. We encountered no issues during the tuning and training phases of RF, and we obtained satisfactory results on the first attempt. In contrast, SVM proved to be considerably slower, leading to problems during training, and in some cases, the process terminated due to memory resource constraints.

Table 9. Information on the model's bestpoint hyperparameters as well as its tuning time

Model	Best point hyperparameters	Tuning time
DT	Maximum number of splits: 36951	8 minutes
	Split criterion: Gini's diversity index	
KNNs	Number of neighbors: 10	21 hours
	Distance weight: Squared inverse	
RF	Ensemble method: Bag	2 hours
	Number of learners: 11	

	Maximum number of splits: 34067	
	Number of predictors to sample: 6	
SVM	Multiclass method: One-vs-One	46 hours
	Box constraint level: 943.1384	
	Kernel scale: 2.8684	
	Kernel function: Gaussian	
	Standardize data: true	

Therefore, if time constraints are a concern and prompt results are desired, RF would be the recommended model. On the other hand, if achieving the highest test accuracy is of utmost importance, SVM would be the optimal choice, despite its longer tuning and training times.

Engineer ethics can become relevant in this study because the decision of which URLs should be classified as malicious or benign and can have important consequences for a user. The engineer must consider the potential consequences of incorrect classifications of URLs, such as a malicious URL being incorrectly classified as benign, which can deceive users into clicking on it and lead to malicious consequences. On the other hand, a benign URL being incorrectly classified as malicious can result in unwanted blocking of users' access to web pages that are necessary for personal use. To address these issues in the classification of URLs using machine learning models, they should be tested and validated to ensure they function as intended. This may involve testing the models with a large number of URLs to ensure their accuracy.

Identifying and blocking malicious URLs can contribute to enhancing cybersecurity and protecting users from cyber threats. This, in turn, can reduce the risk of financial harm and promote a stable and sustainable economy. According to [33], the costs of cybercrime are projected to increase annually by 15% until 2025. It states that the estimated annual cost of cybercrime in 2025 is projected to be 10.5 trillion dollars, which is a comparison to the cost of 3 trillion dollars in 2015.

To conduct the training and testing phases, we explored alternative tools such as Jupyter Notebook, which utilizes Python. However, we encountered certain issues with Jupyter Notebook, such as generating figures like the confusion matrix, which hindered our ability to present and analyze the results effectively. Furthermore, we faced memory space constraints that proved critical during model training. As a result, we resorted to using MATLAB through a remote desktop, a platform shared for many other students at the university. This posed additional challenges, particularly in terms of the extensive training time required for models like KNNs and SVM.

To enhance the study, several improvements can be implemented. Firstly, collecting a larger dataset with a balanced representation of all types of malicious URLs, including Benign, Defacement, Phishing, and Malware, would ensure that the models have an equal opportunity to accurately classify and identify malicious URLs, thus providing a more comprehensive evaluation. Secondly, expanding the range of models examined, such as including Neural Networks, NB, and Gradient Boosting models like XGB and LGB, would allow for the exploration of alternative models that may offer superior performance in the task of malicious URL identification. Lastly, considering additional categories of malicious URLs, like Redirect-URL, Scam-URL, Clickbait-URL, and Drive-by Downloads-URL, would broaden the study's scope and provide insights into the challenges of classifying diverse types of malicious URLs, ultimately contributing to the development of more effective cybersecurity measures.

5 Conclusion

By following the classification process, the results were obtained and compared. The findings demonstrated that the random instance selection method exhibited a shorter elapsed time in selecting data and its models achieved higher accuracies compared to the other two methods, BPLSH and DRLSH. The reason why these two methods couldn't achieve high accuracies may be the imbalanced dataset, which resulted in poor sample selection.

In each of the three datasets, different machine learning models had the highest accuracy, indicating that the number of observations for each category and the URLs selected in each dataset by the instance selection methods played important roles in the accuracies of the models. On average, the SVM model demonstrated the highest average test accuracy compared to the other models, reaching 90.9%. In second place, the RF model achieved an average test accuracy of 90.8%.

The feature that had the most significant impact and played a crucial role in classifying the malicious URLs was "has_http." This feature indicated whether the URLs were not encrypted and had a high probability of being malicious URLs. The feature "has_html" has also played an important role in classifying defacement URLs in this study. It was observed that most of defacement URLs contained the term "html" in their addresses.

In conclusion, this article highlights the importance of having balanced datasets, using suitable instance selection methods, and considering relevant features to achieve accurate classification results for malicious URLs. Further research and exploration in these areas can lead to the development of more effective models and techniques for identifying and classifying online threats.

References

[1] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, and X. Chang, “A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment,” *Computer Communications*, vol. 175, pp. 47–57, Jul. 2021, doi: 10.1016/j.comcom.2021.04.023.

[2] M. Veale and I. Brown, “Cybersecurity,” *Internet Policy Review*, vol. 9, no. 4, Dec. 2020, doi: 10.14763/2020.4.1533.

[3] “What is a Malicious URL and How Do We Protect Against Them?,” May 02, 2023. https://experteq.com/what-is-a-malicious-url-and-how-do-we-protect-against-them/ (accessed May 18, 2023).

[4] D. Ali and S. Frimpong, “Artificial Intelligence, Machine Learning and Process Automation: Existing Knowledge Frontier and Way Forward for Mining Sector,” *Artificial Intelligence Review*, vol. 53, pp. 6025–6042, Dec. 2020, doi: 10.1007/s10462-020-09841-6.

[5] S. Johnson, “AI, Machine Learning, and Ethics in Health Care,” *Journal of Legal Medicine*, vol. 39, pp. 427–441, Oct. 2019, doi: 10.1080/01947648.2019.1690604.

[6] D. Gong, “Top 6 Machine Learning Algorithms for Classification,” *Medium*, Jul. 12, 2022. https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501 (accessed Apr. 07, 2023).

[7] S. H. Ahammad *et al.*, “Phishing URL detection using machine learning methods,” *Advances in Engineering Software*, vol. 173, p. 103288, Nov. 2022, doi: 10.1016/j.advengsoft.2022.103288.

[8] B. Wardman, D. Clemens, J. Wolnski, P. Inc–San Jose, U. States, and S. D.-U. States, “Phorecasting phishing attacks: A new approach for predicting the appearance of phishing websites,” *Cyber-Security and Digital Forensics*, p. 142, 2016.

[9] N. Virvilis, A. Mylonas, N. Tsalis, and D. Gritzalis, “Security Busters: Web browser security vs. rogue sites,” *Computers & Security*, vol. 52, pp. 90–105, Jul. 2015, doi: 10.1016/j.cose.2015.04.009.

[10] D. Sahoo, C. Liu, and S. C. H. Hoi, “Malicious URL Detection using Machine Learning: A Survey.” arXiv, Aug. 21, 2019. doi: 10.48550/arXiv.1701.07179.

[11] “What is a URL? - Learn web development | MDN,” Feb. 23, 2023. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL (accessed Mar. 04, 2023).

[12] A. Singh, N. Thakur, and A. Sharma, “A review of supervised machine learning algorithms,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2016, pp. 1310–1315. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7724478

[13] “Classification Algorithm in Machine Learning - Javatpoint,” *www.javatpoint.com*. https://www.javatpoint.com/classification-algorithm-in-machine-learning (accessed Mar. 04, 2023).

[14] "Classification Process - an overview | ScienceDirect Topics." https://www.sciencedirect.com/topics/engineering/classification-process (accessed Mar. 05, 2023).

[15] A. Jafar and M. Lee, "High-speed hyperparameter optimization for deep ResNet models in image recognition," *Cluster Comput*, May 2021, doi: 10.1007/s10586-021-03284-6.

[16] B. Akinremi, "Best Tools for Model Tuning and Hyperparameter Optimization," *neptune.ai*, Jul. 21, 2022. https://neptune.ai/blog/best-tools-for-model-tuning-and-hyperparameter-optimization (accessed Mar. 12, 2023).

[17] T. Shah, "About Train, Validation and Test Sets in Machine Learning," *Medium*, Jul. 10, 2020. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 (accessed Mar. 13, 2023).

[18] A. K. Baareh, A. Elsayad, and M. Al-Dhaifallah, "Recognition of splice-junction genetic sequences using random forest and Bayesian optimization," *Multimed Tools Appl*, vol. 80, no. 20, pp. 30505–30522, Aug. 2021, doi: 10.1007/s11042-021-10944-7.

[19] M. Aljabri *et al.*, "An Assessment of Lexical, Network, and Content-Based Features for Detecting Malicious URLs Using Machine Learning and Deep Learning Models," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–14, Aug. 2022, doi: 10.1155/2022/3241216.

[20] Y. Li, Z. Yang, X. Chen, H. Yuan, and W. Liu, "A stacking model using URL and HTML features for phishing webpage detection," *Future Generation Computer Systems*, vol. 94, pp. 27–39, May 2019, doi: 10.1016/j.future.2018.11.004.

[21] A. Saleem Raja, R. Vinodini, and A. Kavitha, "Lexical features based malicious URL detection using machine learning techniques," *Materials Today: Proceedings*, vol. 47, pp. 163–166, Jan. 2021, doi: 10.1016/j.matpr.2021.04.041.

[22] "Habib_Mrad-Detection Malicious URL Using ML Models." https://kaggle.com/code/habibmrad1983/habib-mrad-detection-malicious-url-using-ml-models (accessed Apr. 07, 2023).

[23] U. S. D. R, A. Patil, and Mohana, "Malicious URL Detection and Classification Analysis using Machine Learning Models," in *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, Jan. 2023, pp. 470–476. doi: 10.1109/IDCIoT56793.2023.10053422.

[24] "What Is MATLAB?" https://se.mathworks.com/discovery/what-is-matlab.html (accessed Apr. 05, 2023).

[25] "Malicious URLs dataset." https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset (accessed May 16, 2023).

[26] M. Aslani and S. Seipel, "Efficient and decision boundary aware instance selection for support vector machines," *Information Sciences*, vol. 577, pp. 579–598, Oct. 2021, doi: 10.1016/j.ins.2021.07.015.

[27] M. Aslani and S. Seipel, "A fast instance selection method for support vector machines in building extraction," *Applied Soft Computing*, vol. 97, p. 106716, Dec. 2020, doi: 10.1016/j.asoc.2020.106716.

[28] H. Sharma and S. Kumar, "A Survey on Decision Tree Algorithms of Classification in Data Mining," *International Journal of Science and Research (IJSR)*, vol. 5, Apr. 2016, [Online]. Available: https://www.ijsr.net/archive/v5i4/NOV162954.pdf

[29] A. Adeyemi and A. Mosavi, "Domain Driven Data Mining - Application to Business," *International Journal of Computer Science Issues (impact factor: 0.3418)*, Jan. 2010, [Online]. Available: https://www.researchgate.net/publication/249313178_Domain_Driven_Data_Mining_-_Application_to_Business

[30] M. S. Alam and S. T. Vuong, "Random Forest Classification for Detecting Android Malware," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Aug. 2013, pp. 663–669. doi: 10.1109/GreenCom-iThings-CPSCom.2013.122.

[31] A. Saini, "Support Vector Machine(SVM): A Complete guide for beginners," *Analytics Vidhya*, Oct. 12, 2021. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/ (accessed Mar. 12, 2023).

[32] P. Nellihela, "What is K-fold Cross Validation?," *Medium*, Nov. 01, 2022. https://towardsdatascience.com/what-is-k-fold-cross-validation-5a7bb241d82f (accessed May 22, 2023).

[33] C. Woodun, "Fortinet Should Benefit From The SolarWinds-Microsoft Hack (NASDAQ:FTNT) | Seeking Alpha," Dec. 22, 2020. https://seekingalpha.com/article/4395774-fortinet-should-benefit-from-solarwinds-microsoft-hack, https://seekingalpha.com/article/4395774-fortinet-should-benefit-from-solarwinds-microsoft-hack (accessed Mar. 14, 2023).

Appendix A. RF and DT in random dataset

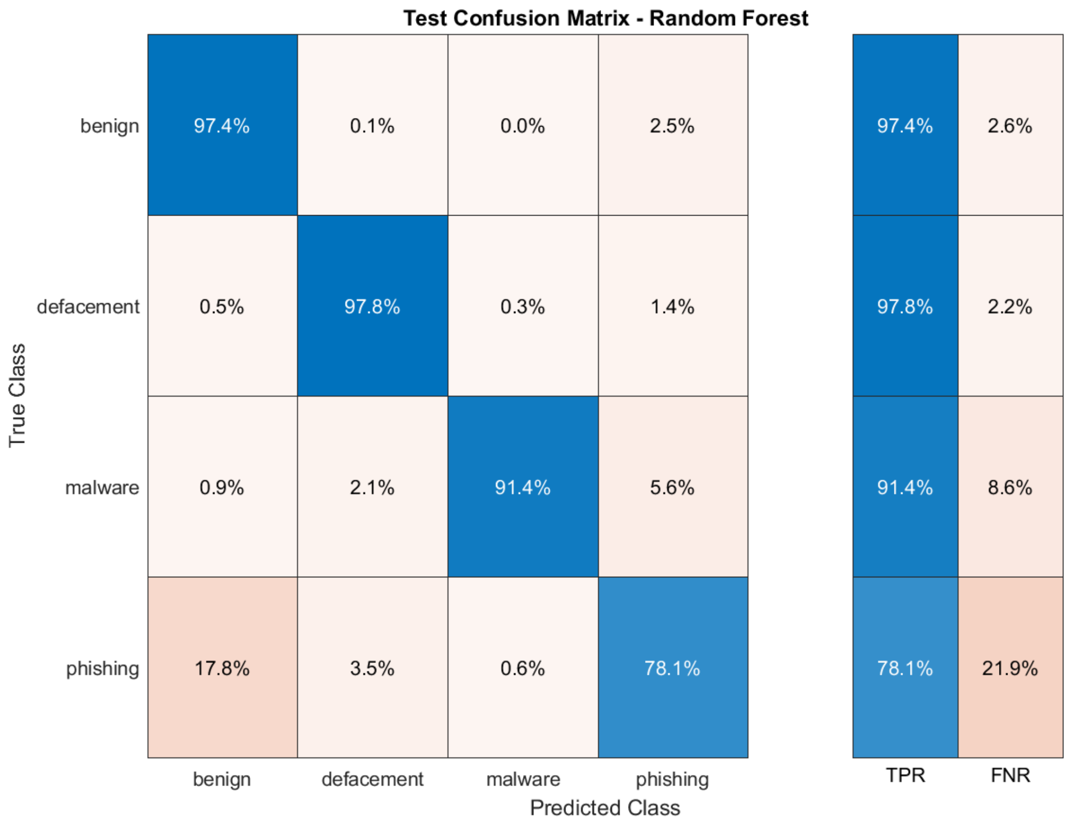


Figure A.1: Test confusion matrix for RF model.

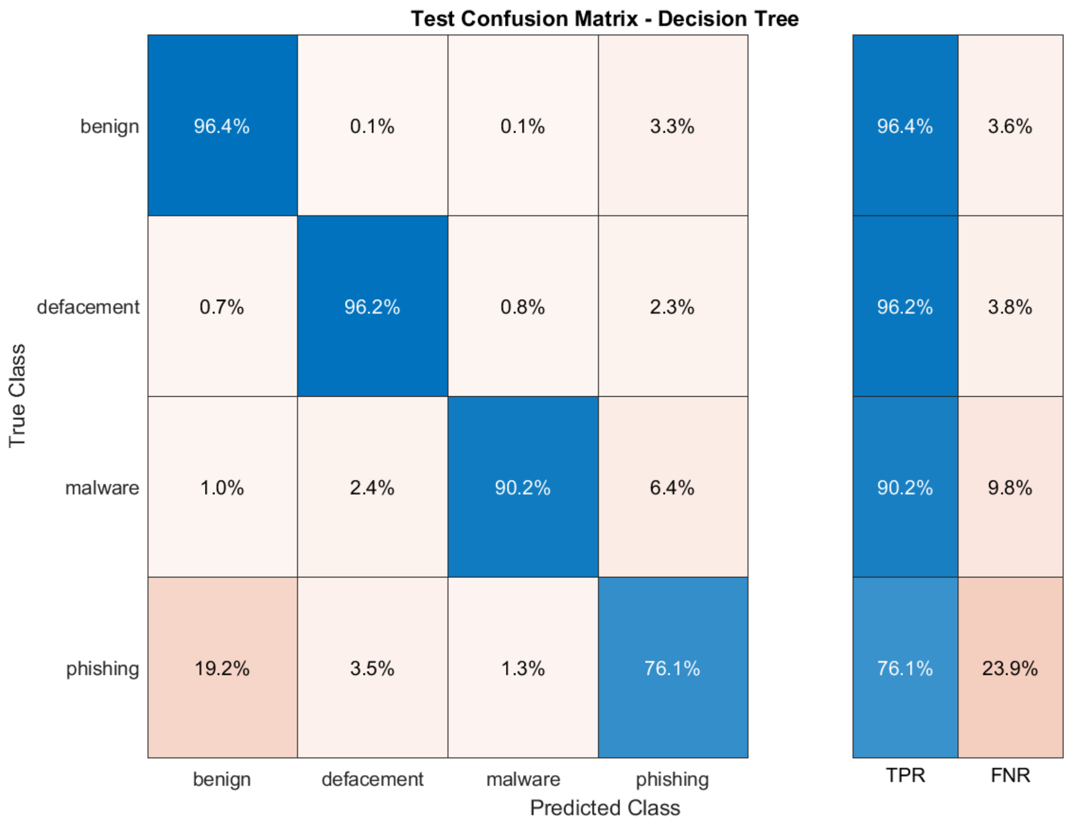


Figure A.2: Test confusion matrix for DT model.

Appendix B. SVM and KNNs in random dataset

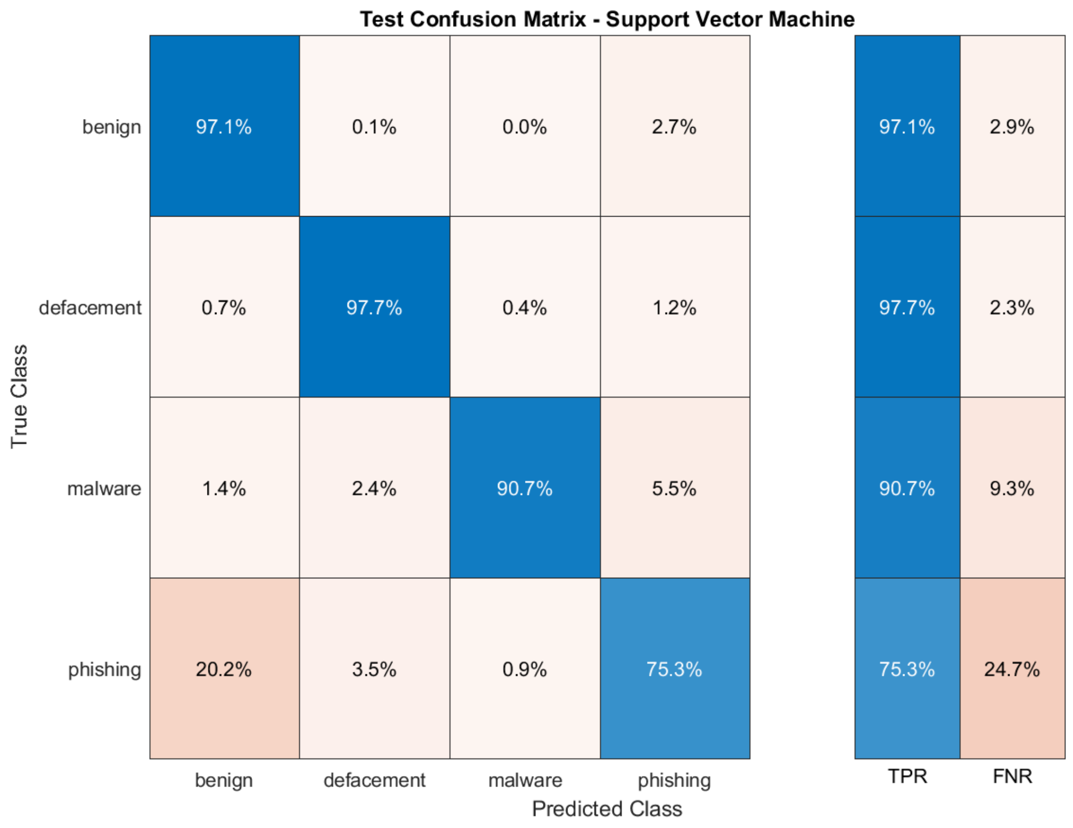


Figure B.1: Test confusion matrix for SVM model.

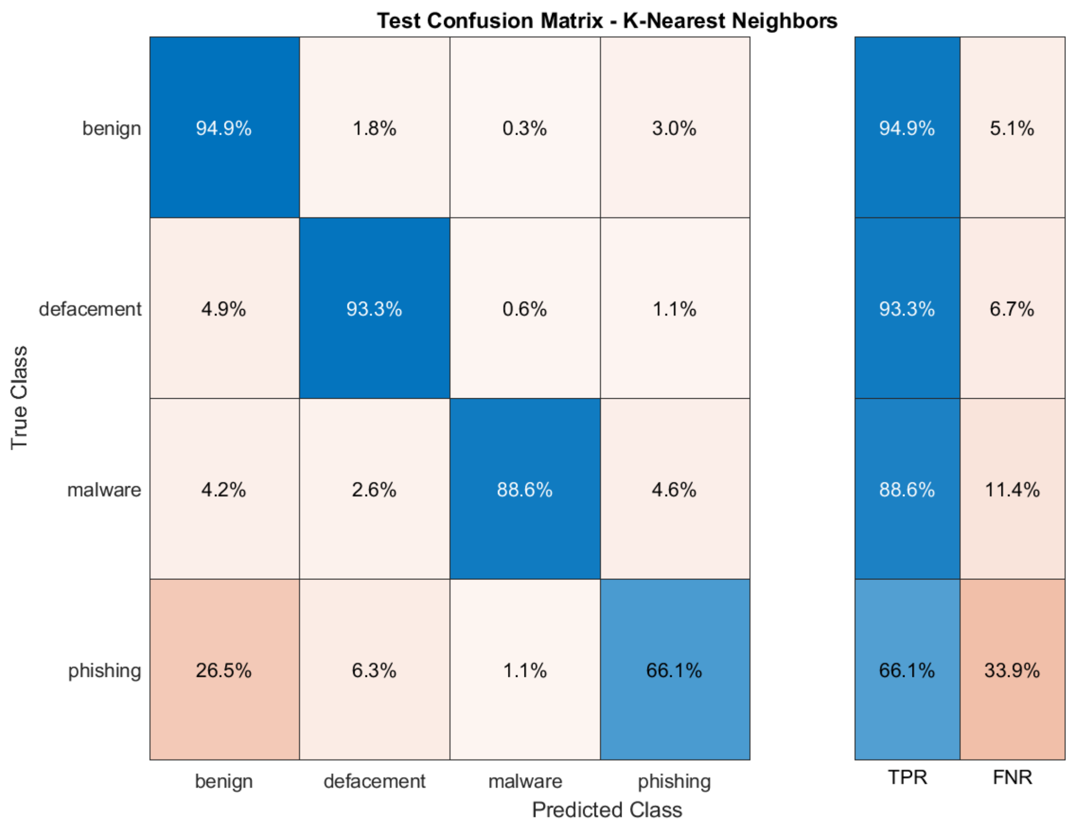


Figure B.2: Test confusion matrix for KNNs model.

Appendix C. RF and DT in DRLSH dataset

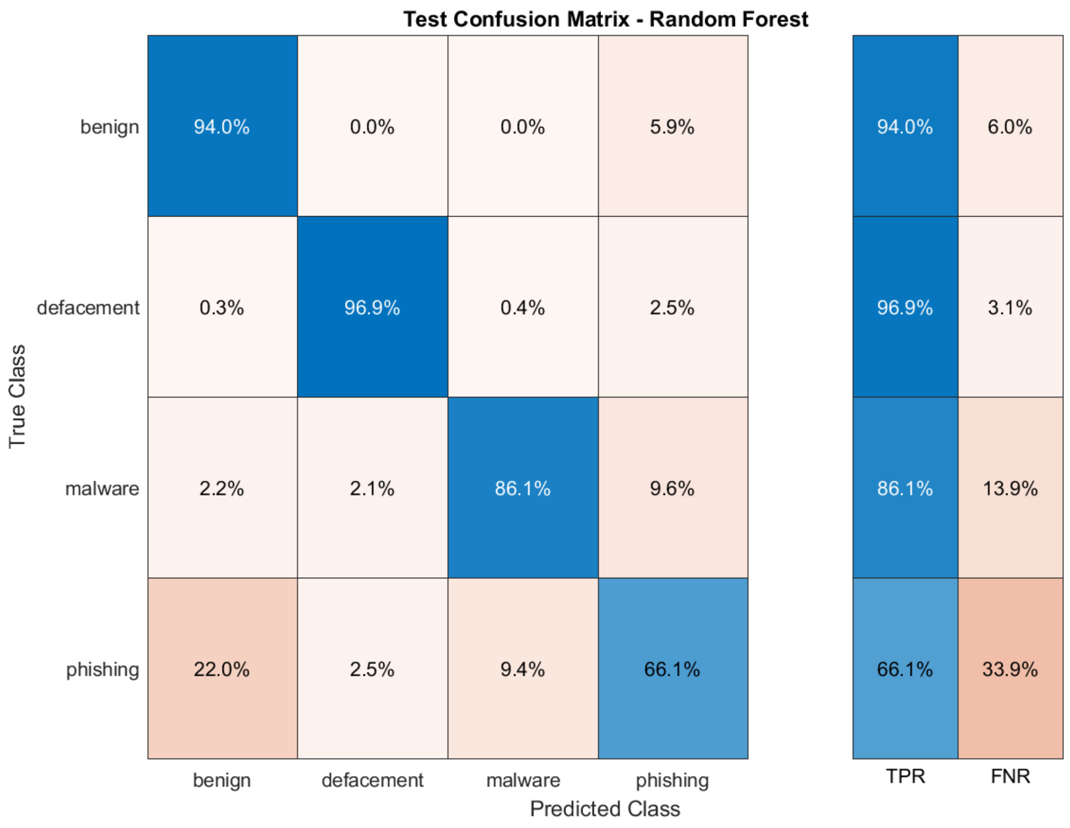


Figure C.1: Test confusion matrix for RF model.

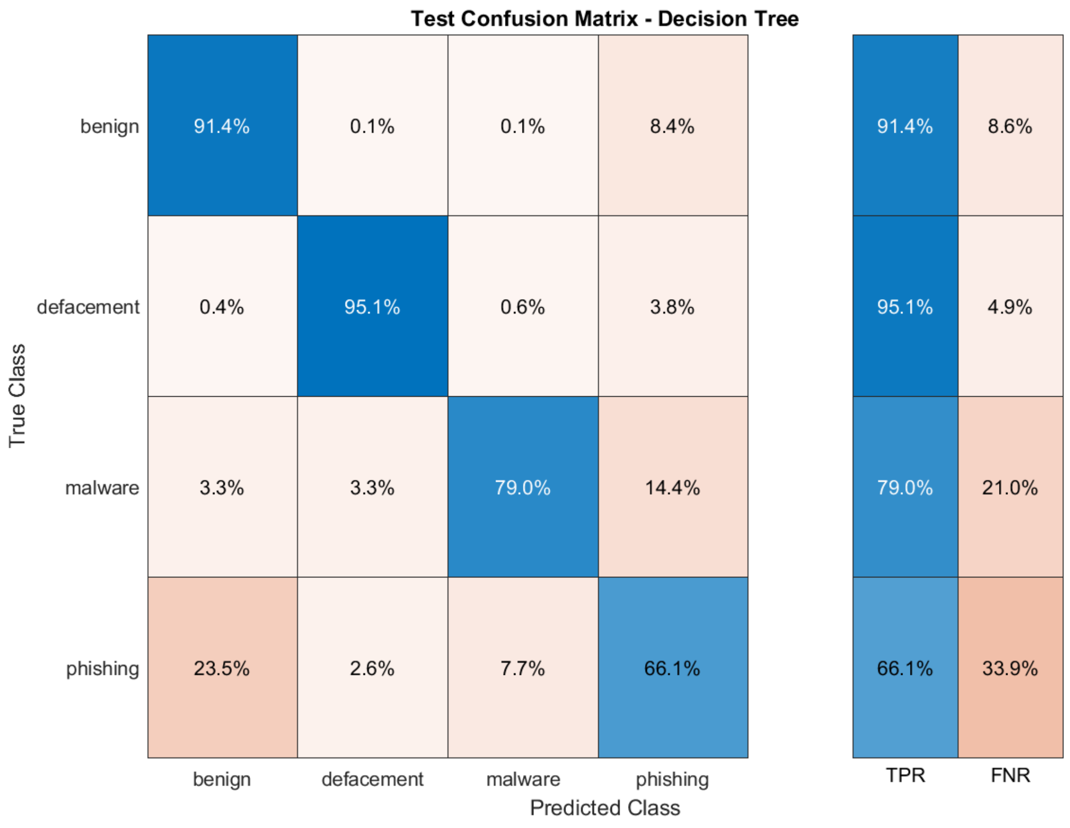
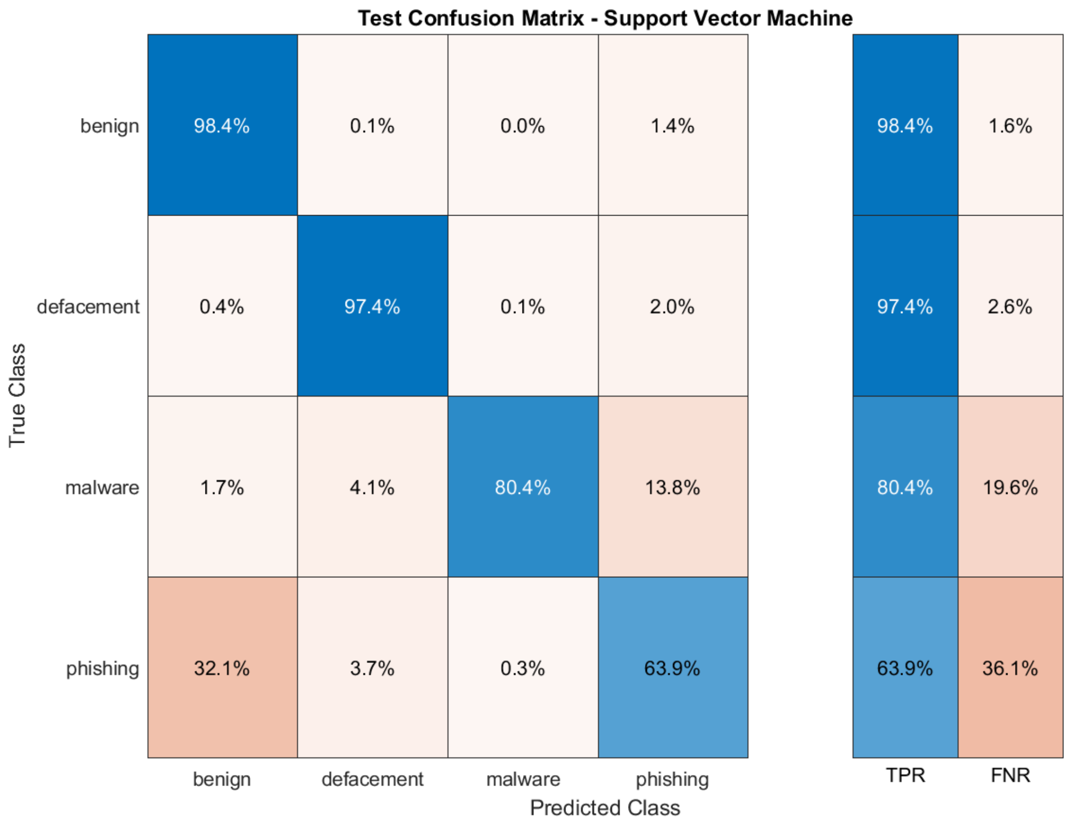


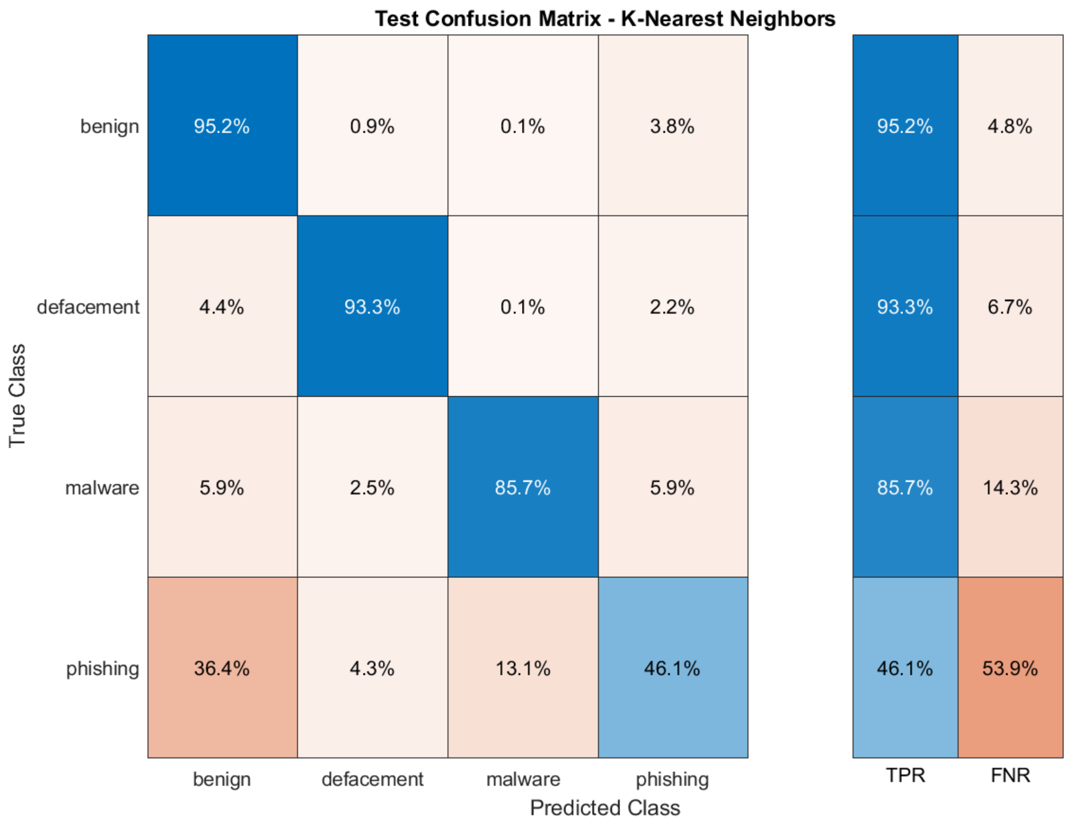
Figure C.2: Test confusion matrix for DT model.

Appendix D. SVM and KNNs in DRLSH dataset



True Class	benign	defacement	malware	phishing		TPR	FNR
benign	98.4%	0.1%	0.0%	1.4%		98.4%	1.6%
defacement	0.4%	97.4%	0.1%	2.0%		97.4%	2.6%
malware	1.7%	4.1%	80.4%	13.8%		80.4%	19.6%
phishing	32.1%	3.7%	0.3%	63.9%		63.9%	36.1%

Figure D.1: Test confusion matrix for SVM model.



True Class	benign	defacement	malware	phishing		TPR	FNR
benign	95.2%	0.9%	0.1%	3.8%		95.2%	4.8%
defacement	4.4%	93.3%	0.1%	2.2%		93.3%	6.7%
malware	5.9%	2.5%	85.7%	5.9%		85.7%	14.3%
phishing	36.4%	4.3%	13.1%	46.1%		46.1%	53.9%

Figure D.2: Test confusion matrix for KNNs model.

Appendix E. RF and DT in BPLSH dataset

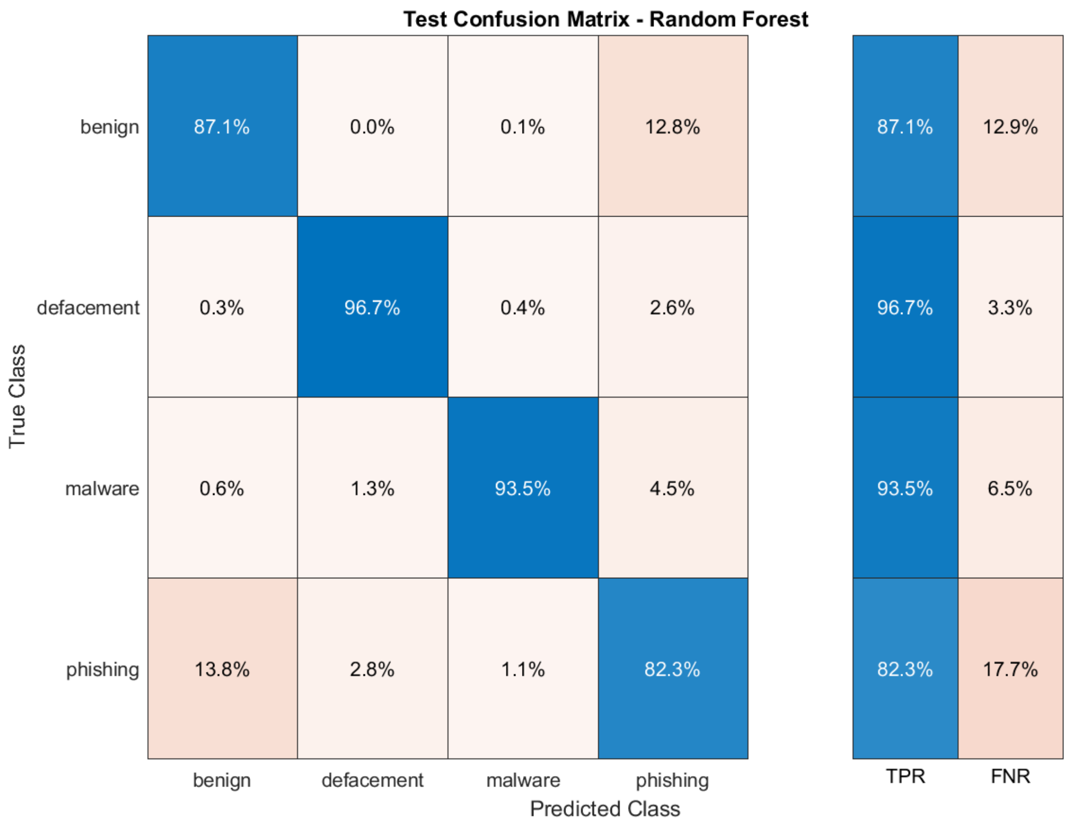


Figure E.1: Test confusion matrix for RF model.

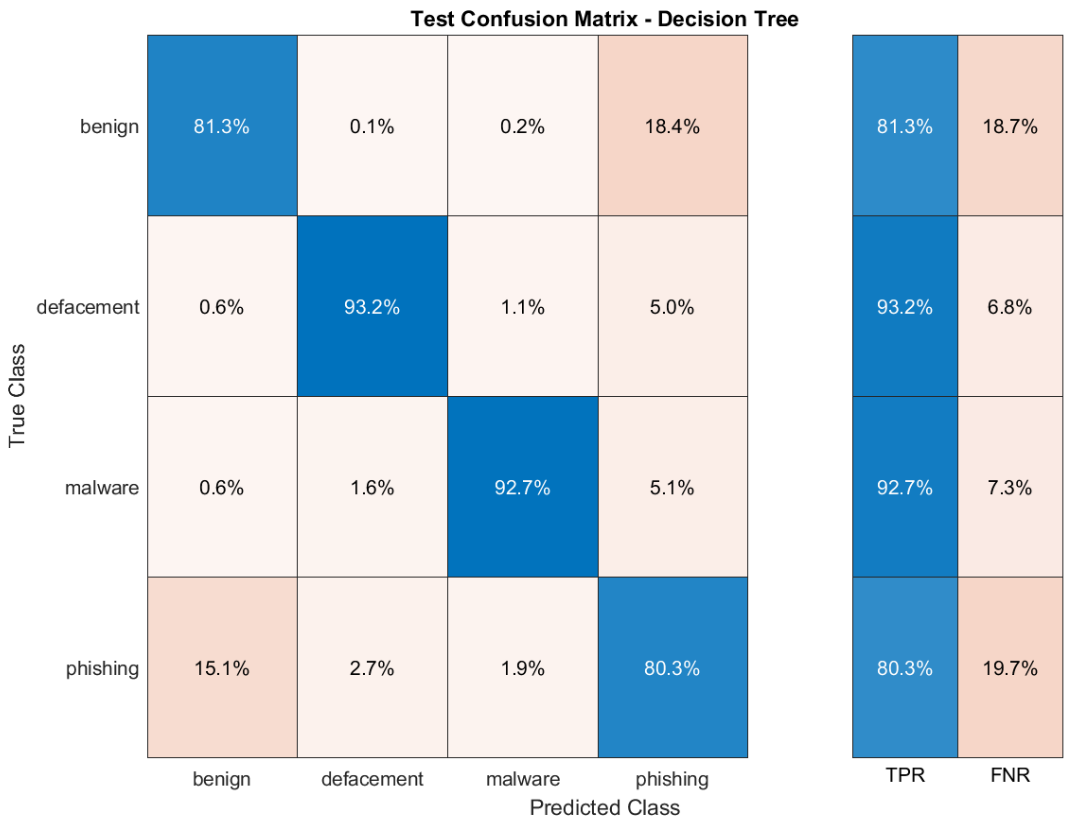


Figure E.2: Test confusion matrix for DT model.

Appendix F. SVM and KNNs in BPLSH dataset

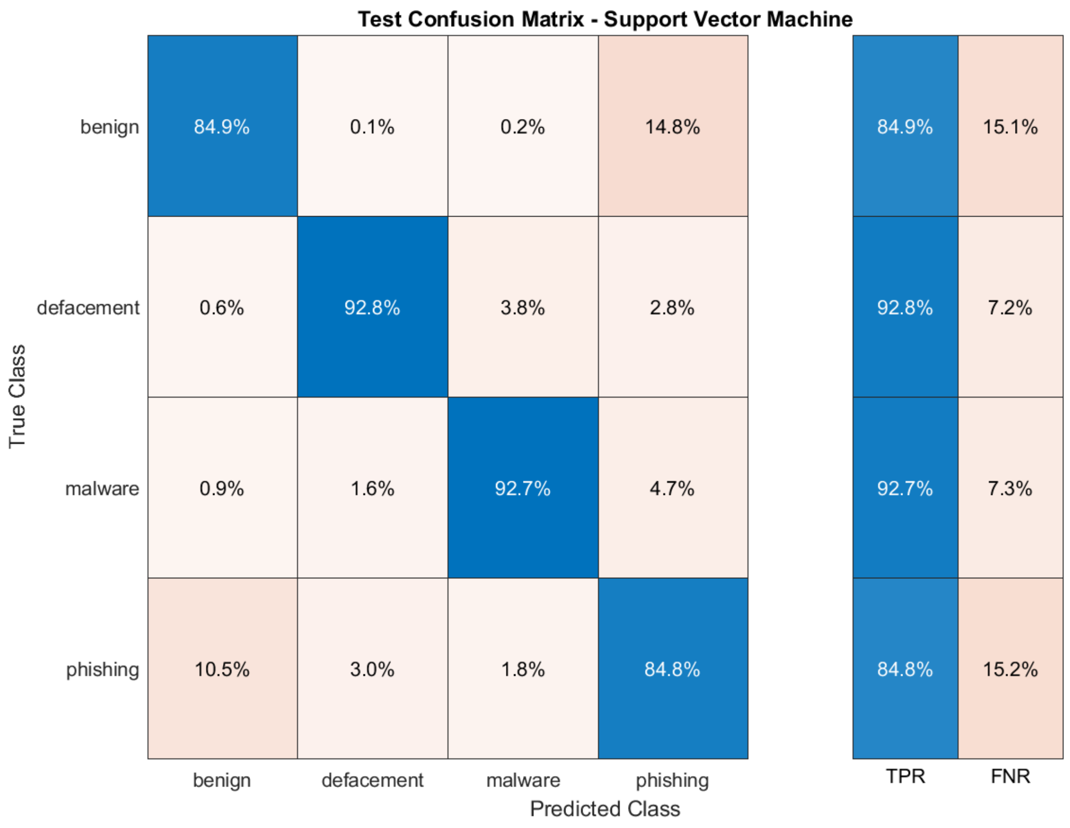


Figure F.1: Test confusion matrix for SVM model.

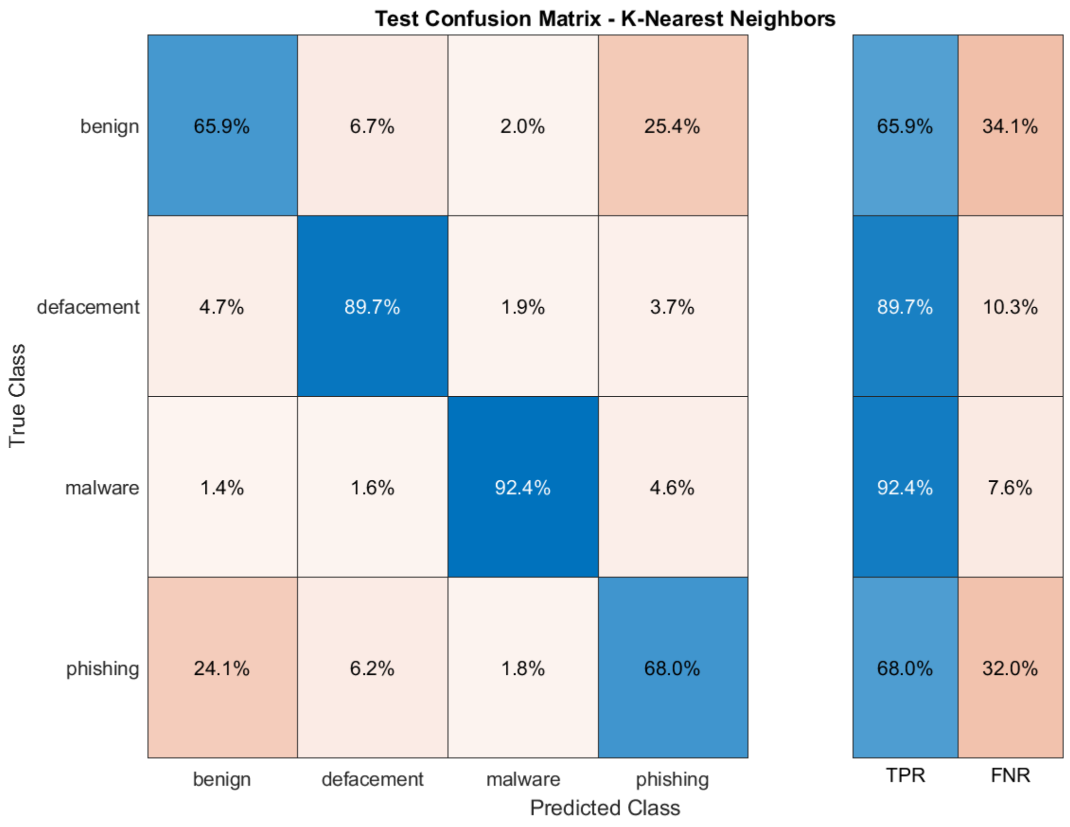


Figure F.2: Test confusion matrix for KNNs model.