THOMAS ADEWUMI UNIVERSITY, OKO, KWARA STATE
Science | Technology | Medicine

Thomas Adewumi University Journal of Innovation, Science and Technology (TAU-JIST)

ISSN: 3043-503X

RESEARCH ARTICLE

ASSESSMENT OF MACHINE LEARNING MODELS FOR THE CLASSIFICATION OF MALICIOUS URLS

[1]*Olofinlade Samuel Oluwapelumi,* [2]*Ogunjimi Olalekan L. A.* [3]*Adebayo Oni Patric and* [4]*Sarumi Jeremial A.*

[1]*University of Ilorin, Dept. of Accounting and Finance, Ilorin, Kwara State, Nigeria.*

[2]*Phoenix University, Dept. of Computer and Software Engineering, Agwada, Nasarawa State, Nigeria*

[3]*Phoenix University, Dept. of Mathematics and Statistics, Agwada, Nasarawa State, Nigeria.*

[4]*Ladoke Akintola University of Technology, Dept. of Computer Science, Ogbomoso, Oyo State, Nigeria.*

Correspondence author Email: olofinlade.soo@unilorin.edu.ng.

ARTICLE DETAILS

Article History:
Received 02 July 2024
Accepted 05 October 2024
Available online 10 December 2024

ABSTRACT

Identifying safe websites can be a challenging task as millions of new websites are created every day. Cybersecurity is the practice of safeguarding individuals and organizations against online threats. Phishing attacks are one of the many tactics used by cybercriminals to deceive people into revealing their personal information. In 2022, over 74,000 phishing attacks were reported in Australia alone, resulting in financial losses exceeding $24 million. In various domains such as the detection of financial fraud, and cancer, and the development of chatbots, artificial intelligence (AI) and machine learning have proven to be useful tools. Support Vector Machines and Random Forest are often employed as machine learning models for classification tasks. With the rise in cybercrime, machine learning is crucial for identifying fraudulent URLs, both new and old. The study comparing different machine learning models for classifying malicious URLs found that the Random Forest model achieved the highest accuracy, followed by the Support Vector Machine. Key findings emphasized the importance of balanced datasets, appropriate instance selection methods, and the feature "has HTTP" in achieving accurate results. Further research is suggested to explore additional models and categories of malicious URLs for enhanced cybersecurity measures. In summary, Random Forest outperforms other models in classifying malicious URLs, and balanced datasets and relevant features are crucial for optimal performance. Around 650,000 URLs from Kaggle were used in the dataset for this investigation. Malware, benign URLs, defacement, and phishing were the four categories that made up the dataset. Instance selection techniques (DRLSH, BPLSH, and random selection) written in MATLAB were used to construct three datasets, each including approximately 170,000 URLs. We used SVM, DT, KNNs, and RF as examples of machine-learning models. In order to train the machine learning models on the malicious URL datasets, the study employed these instance selection techniques. Then, using 16 features and one output feature, it assessed the models' performance. During the hyperparameter tuning procedure, four models with various hyperparameter settings were trained using the training dataset. For each model, the ideal hyperparameter was determined via Bayesian optimization. The system of classification.

KEYWORDS

Machine Learning, Cyber Security, Classification, Malicious URL, Instance Selection.

Quick Response Code	Access this article online	
	Website: https://journals.tau.edu.ng/index.php/tau-jist	**DOI:** https://doi.org/10.5281/zenodo.15022046

Cite the Article ***Olofinlade Samuel Oluwapelumi, Ogunjimi Olalekan L. A. Adebayo Oni Patric and Sarumi Jeremial A.*** *(2024). Assessment of Machine Learning Models or The Classification of Malicious Urls*

1. INTRODUCTION

Every day, millions of new websites are launched, collecting user data via login functions. The enormous number of networks makes it difficult to evaluate which ones are safe and reliable. Cybersecurity can be described as a set of technologies or procedures used to safeguard businesses and individuals from cyber threats.

Cybercriminals utilize a variety of methods to trick internet users into disclosing sensitive and personal information, one of which is via malicious URLs, or hyperlinks. Interacting with these links exposes users to a variety of consequences, ranging from the compromise of critical information to being a major target for cyberattacks. The Australian Competition and Consumer Commission documented a total of 74.567 reported cases of phishing attacks.

1.1 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is the art of computing in which a machine can accomplish specified tasks without the need for explicit programming. AI aspires to be as intelligent as the human brain. AI has proven effective in several domains, including detecting and diagnosing malignant tissues faster than doctors, detecting illegal schemes involving money transactions, and constructing chatbots that can talk and understand language and become helpful.

1.1.1 MACHINE LEARNING

Machine learning is an artificial intelligence (AI) application that uses statistical methods and algorithms to detect patterns, structures, and categories in big datasets. There are two forms of machine learning: supervised and unsupervised. Supervised learning uses pre-defined data to provide access to the model's inputs and outputs, whereas unsupervised learning does not. There is also a subset of machine learning called deep learning, which mimics the human brain with several layers.

There are several effective machine learning models for classification, including Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbors (KNNs, and Naive Bayes (NB).

1.2 PROBLEM DESCRIPTION

With the increasing use of the internet, new forms of data theft have emerged, known as cybercrime. This involves breaching privacy through computer use and accessing confidential and sensitive information unlawfully. Phishing is one of the most common techniques used by attackers, where they try to trick the victim into revealing sensitive information, which can have devastating consequences. Another technique used by hackers is defacement, where they manipulate the underlying code to modify web page content. This form of attack is used to exploit an organization's website. The Rivest-Shamir-Adleman (RSA) internet fraud report of 2014 revealed that approximately 450,000 phishing websites resulted in a loss of $5.9 billion. A blacklist of known URLs is used for searching. This should not be too complicated.

1.3 PURPOSE

This study aims to examine and contrast the effectiveness of three distinct instance selection techniques and four different machine learning models to categorize malevolent URLs. This study answers the following research questions:

What are the key features that play a significant role in classifying malevolent URLs in the dataset?

Which machine learning models and instance selection methods exhibit better performance on the dataset?

2. THEORY

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.

Figure 1. URL anatomy.

2.2 MACHINE LEARNING

The most prevalent machine learning technique for categorizing dangerous URLs is classification, which is a type of supervised machine learning because it is based on prior knowledge.

2.2.1 CLASSIFICATION

Classification is a type of supervised learning that involves the process of presenting and categorizing data based on previous training data. This method requires training the algorithm with a pre-categorized dataset and then applying it to new data by recognizing patterns in the training data. The classification process plays a vital role in categorizing data based on its unique features.

During the preparation phase, a machine learning classification model requires appropriate training, which includes selecting a suitable dataset, removing any 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 separate dataset that doesn't include samples used during the training phase. This allows for an accurate assessment of the model's performance with new data.

2.2.2 HYPERPARAMETERS

Hyperparameters are settings that impact the training of machine learning models. By adjusting these parameters, the developer can improve the model's performance, leading to higher accuracy and shorter learning times. Hyperparameter optimization is a process that aims to optimize the hyperparameters of a model to achieve the best possible performance on a dataset. Machine learning involves training, validation, and testing processes to evaluate the model's performance and avoid overfitting, which can occur when the model over-emphasizes each data point in the training dataset. This can cause the model to perform poorly when applied to new data. Various techniques can be employed to address this issue and enhance the model's performance with new data.

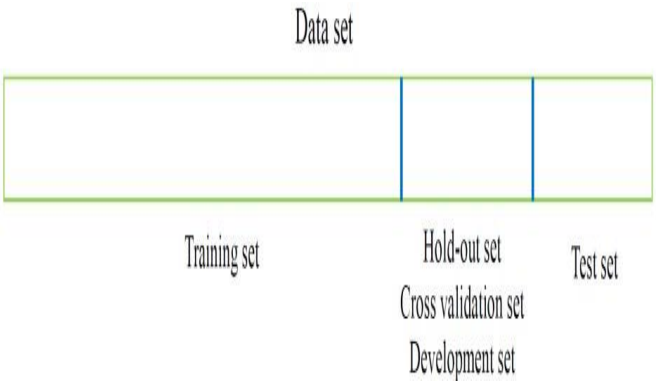

Figure 2. Training, validation, and testing processes. Source [1]

The training data is utilized to build the model. The validation data is a different dataset that was not used during model training. Validation data is used to evaluate the model's performance. After training and validation, the data is tested to see how well the model performs on new datasets.

Bayesian optimization is a powerful technique for optimizing model settings. It leverages a surrogate model that's based on a range of hyperparameter combinations for the model. The algorithm selects the next point to evaluate the objective function, f(x), by using the surrogate model to determine the most effective optimization step. This approach significantly speeds up the process and reduces the number of attempts required. There are three primary measures of improvement: "likely to improve", "expected improvement", and "lower confidence limit". The Gaussian process is the most commonly used model in Bayesian optimization.

The Gaussian process is a probabilistic regression model that's used to understand and predict unknown functions, f(x). It's composed of a mean function, m(x), and a kernel function, k(x, xs). The mean function provides an initial estimate or baseline for what the unknown function, f(x), could look like. This provides us with a starting point to make predictions and understand the general behavior of the data. The kernel function, on the other hand, 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 enables us to make predictions based on this information.

2.3 PREVIOUS RESEARCH

Various studies have been conducted on detecting and classifying malicious URLs using machine learning models. These studies have resulted in different levels of accuracy and have contributed to a better understanding of effective techniques for identifying and classifying malicious URLs.

For instance, M.Sc. Aljabri conducted a comprehensive analysis using over one million URLs that had various attributes such as 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. The study used twelve features and involved four steps. The dataset was first analyzed to understand its existing features, and useful features were then extracted. Alternative machine learning models such as NB and RF were used to prepare inputs for machine learning models and analyze the results, and the NB model was found to be the most accurate with an accuracy of 95%.

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

Another study by A. Saleem Raja used a dataset of about 66,000 URLs. In this study, 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%.

S.H. Ahmed used a dataset of 3000 URLs, where 1500 were malicious and 1500 were benign. The study aimed 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 used 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.

Lastly, A. Patil used machine learning to detect malicious URLs in a study that used a total of 651,191 URLs categorized as malware, defacement, benign, and phishing. The study used models like XGB, LGB, and RF to detect and classify malicious URLs. Feature extraction was carried out to facilitate better decision-making by the model, such as the use of IP addresses and the calculation of letters, digits, and non-alphanumeric characters. The result showed that RF had the highest accuracy of over

91%. In summary, these studies have contributed significantly to the advancement of machine-learning techniques in detecting and classifying malicious URLs.

3. PROCESS AND RESULTS

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 hurt 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.

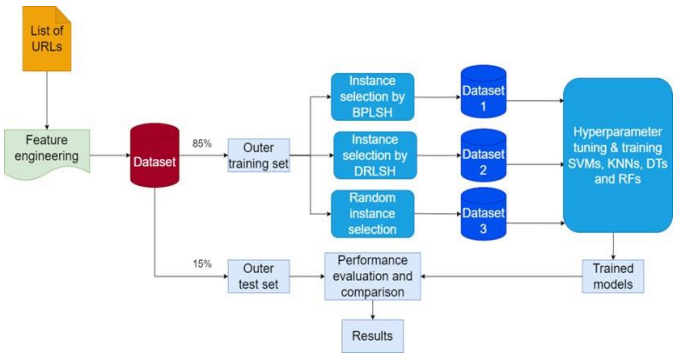

Figure 3. Workflow of the study

3.1.1 DATA COLLECTION

A dataset from Kaggle's data analysis platform, consisting of approximately 650,000 URLs, was utilized. The dataset included four categories of URLs: phishing, defacement, malware, and benign URLs. It contained two columns - one for the URL link and another for the URL type. However, the dataset was found to be imbalanced, as 66% of the data was for benign URLs, while the rest was spread across the remaining categories. The imbalanced dataset was used to train and test the models and methods to evaluate their accuracy and performance. This approach was selected to assess how well the models and instance selection methods handle the challenges posed by imbalanced data.

To generate the datasets, three instance selection methods were employed in MATLAB. Three datasets, each consisting of around 170,000 URLs, were created from the total dataset.

3.1.1.1 RANDOM

To create a dataset of random data, the randperm function in MATLAB was utilized. Approximately 170,000 URLs were randomly selected from the training dataset using this function. This instance selection method was the most efficient way to reduce the data, taking only a few seconds.

3.1.1.2 BPLSH

The BPLSH data selection method, which was developed by M. Aslani and S. Seipel, is used to identify instances that are close to the boundaries of two categories while removing non-essential instances that are distant from these boundaries. This process is illustrated in Figure 4. In the study's dataset, which includes four categories (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 and took more than 1.5 hours

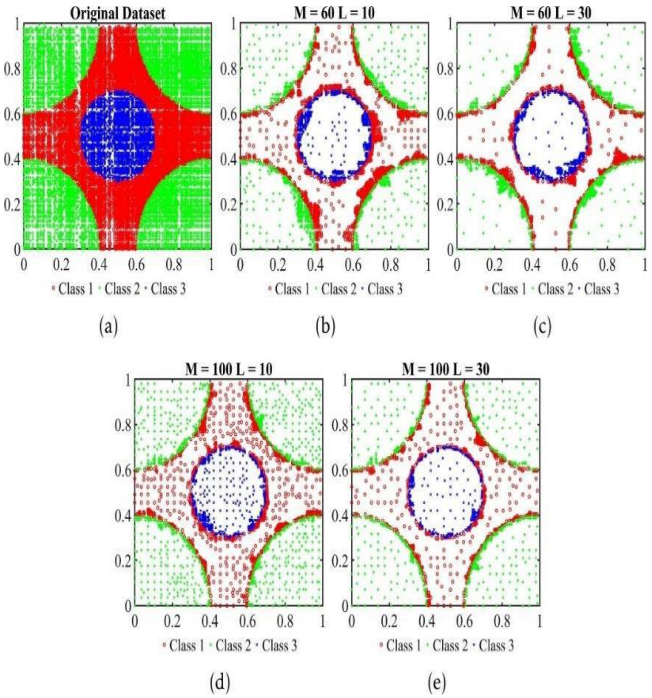

Figure 4. BPLSH process

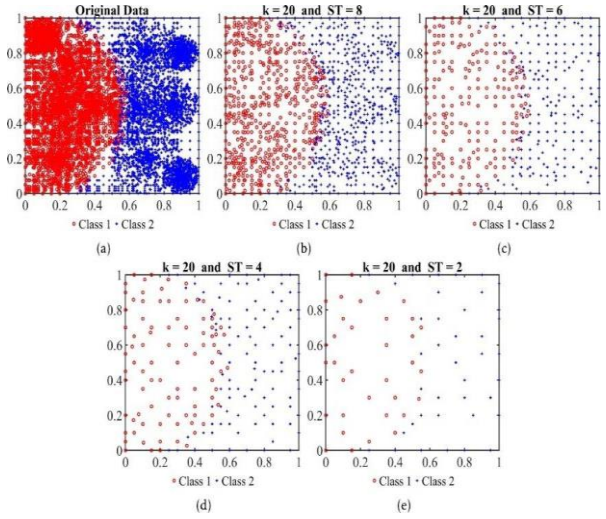

Figure 5. DRLSH process

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).

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

In this study, we have selected four models for comparison: Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNNs), and Support Vector Machine (SVM). The selection of these models was based on their popularity and prior usage in the classification area. The Decision Tree (DT) model creates a tree-like structure to make decisions based on feature values. The Random Forest (RF) model combines multiple decision trees for improved accuracy. The K-Nearest Neighbors (KNNs) model assigns class labels based on the majority of the k's nearest neighbors. Finally, the Support Vector Machine (SVM) model creates a hyperplane to separate different classes in high-dimensional space.

3.1.3.1 DECISION TREE

The decision tree model is a popular tool used in data science and machine learning to classify data based on its features. This model creates a tree structure comprising decision nodes that represent different paths in the decision-making process. This approach helps to identify the key decision factors and how they impact the outcome [28, p. 2094].

To optimize the decision tree model, Bayesian optimization was used to identify the best combination of hyperparameters that would maximize accuracy. The maximum number of decision nodes in the tree was set to one to reduce the tree's complexity. The split criterion was used to measure the quality of the various divisions in the tree.

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

The Gini diversity index is a way to measure segregation parameters in decision tree models. In this method, the proportion of group members belonging ρ_i to a particular class is denoted by It is used to evaluate how accurately the tree divides the data into different classes.

3.1.3.2 K-NEAREST NEIGHBOURS

K-nearest neighbors (KNNs) is a machine learning model for classifying data based on the 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.

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 a 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.

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. 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

3.2.1 HYPERPARAMETERS TUNING

In the hyperparameters tuning process, the training dataset was utilized to train those four models for each of its hyperparameters and try different methods to find the best settings for a model's 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 the 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 uses 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 better classification. Number of learners which refers to the number of decision trees that should be created in the random forest model. A maximum number of splits, which refers to the maximum number of divisions that can be found in the tree, 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 specifies how many predictors are randomly selected at each split point in the tree.

The SVM model's optimal hyperparameters were determined as follows: the 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. The 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. The 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. Standardized 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).

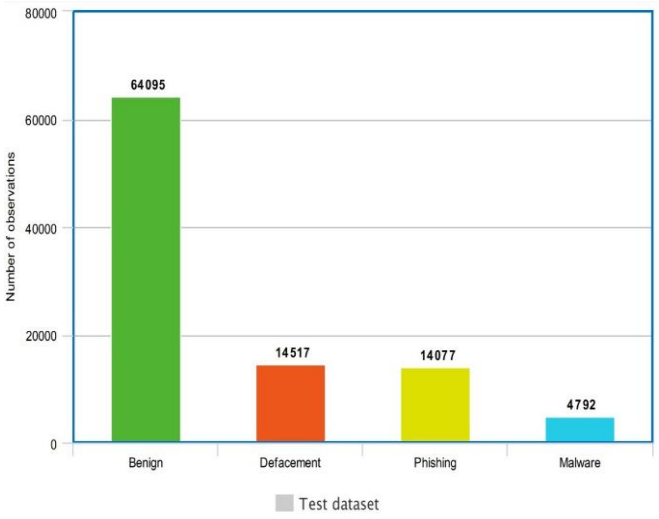

Figure 6. Number of observations for each category in the test dataset.

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 was randomly selected around 170,000 samples from the training dataset. As a result, most of the selected samples belonged to benign URLs (Figure 7).

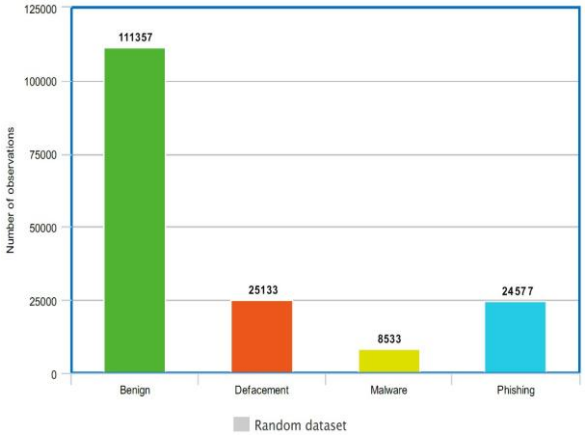

Figure 7. Number of observations for each category in the random dataset.

RF exhibits the highest accuracy compared to the other models (Table 3). Furthermore, considering the relatively short training time, this model can be considered 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

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

The 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).

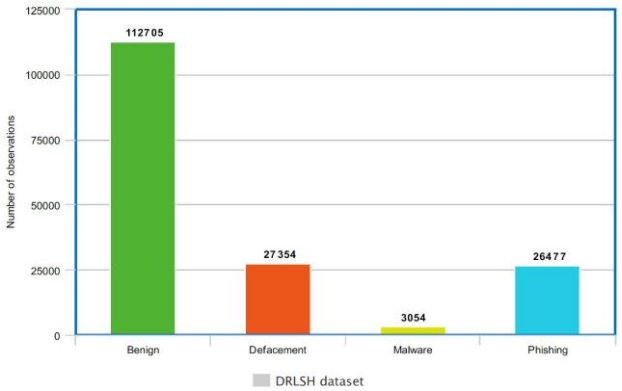

Figure 8. Number of observations for each category in the DRLSH dataset.

The 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.

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).

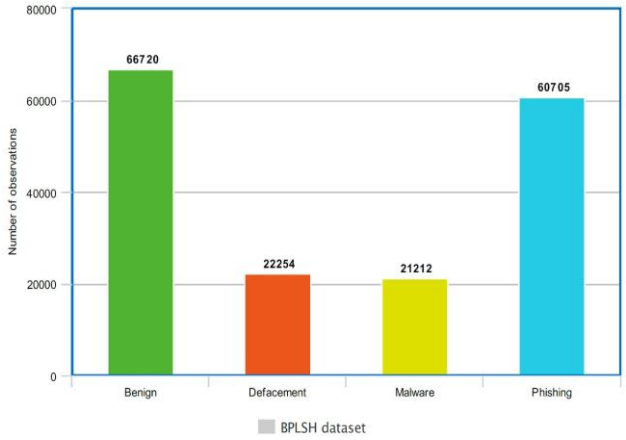

Figure 9. Several observations for each category in the BPLSH dataset.

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 results of the confusion matrices for the Random Forest (RF) and Decision Tree (DT) models show that they accurately predicted defacement and malware URLs, as shown in Appendix E. However, the accuracies for the other categories were below 90%.

Similarly, the confusion matrix for the Support Vector Machine (SVM) model indicates an accuracy of around 92% in predicting defacement and malware URLs, whereas the accuracies for the other two categories were below 90%.

In contrast, the K-Nearest Neighbor (KNN) model performed better than the other models in predicting malware URLs, achieving an accuracy of 92.4%, as shown in its confusion matrix.

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 a protocol are not encrypted and have a high probability of being malicious URLs. 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.

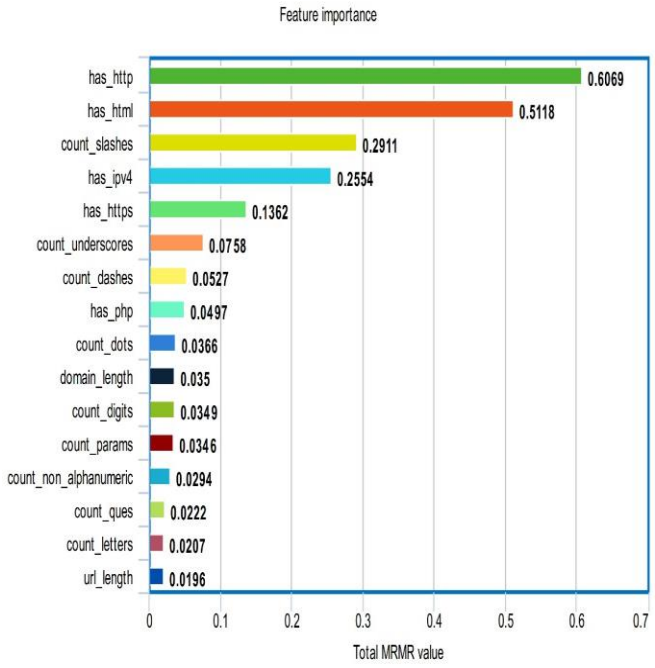

Figure 10. Feature importance

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 the 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 the 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 the 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 expected 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 accuracy.

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 favourable 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 best point hyperparameters as well as its tuning time

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

I have reviewed the text and made some minor corrections to make it clearer. Here's the revised text: 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 are relevant to this study because the decision of which URLs should be classified as malicious or benign can have important consequences for a user. The engineer must consider the potential consequences of incorrect classifications, such as a malicious URL being incorrectly classified as benign, which can deceive users into clicking on it and lead to malicious consequences. Conversely, 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, the machine learning models 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 enhance cybersecurity and protect users from cyber threats. This 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 with 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

After following the classification process, the obtained results were compared. The analysis showed that the random instance selection method was the most efficient in selecting data, resulting in higher accuracies compared to the other two methods, BPLSH and DRLSH. The poor sample selection of these two methods may be due to the imbalanced dataset.

Different machine learning models achieved the highest accuracy in each of the three datasets. The number of observations for each category and the URLs selected in each dataset by the instance selection methods played significant roles in the accuracies of the models. On average, the SVM model demonstrated the highest test accuracy, reaching 90.9%, followed by the RF model which achieved an average test accuracy of 90.8%.

The most significant feature that 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" also played an important role in classifying defacement URLs. It was observed that most of the defacement URLs contained the term "HTML" in their addresses. In conclusion, this article emphasizes the importance of having balanced datasets, using appropriate 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.

6. 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 (2020), "Artificial Intelligence, Machine Learning and Process Automation: Existing Knowledge Frontier and Way Forward for Mining Sector," *Artificial Intelligence Review*, vol. 53, pp. 6025–6042, 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, 022. 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, "Forecasting 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/enS/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)