electronics

MDPI

Article

Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection

Saleem Raja Abdul Samad [1], Sundarvadivazhagan Balasubaramanian [2], Amna Salim Al-Kaabi [1], Bhisham Sharma [3], Subrata Chowdhury [4], Abolfazl Mehbodniya [5,*], Julian L. Webber [5] and Ali Bostani [6]

1 IT Department, University of Technology and Applied Sciences, Shinas 324, Oman
2 IT Department, University of Technology and Applied Sciences, Al-Musannah 314, Oman
3 Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura 140401, Punjab, India
4 Department of Computer Science and Engineering, Sreenivasa Institute of Technology and Management Studies, Chittoor 517127, Andra Pradesh, India
5 Department of Electronics and Communication Engineering, Kuwait College of Science and Technology (KCST), Doha Area, 7th Ring Road, Kuwait 7207, Kuwait
6 College of Engineering and Applied Sciences, American University of Kuwait, Salmiya 20002, Kuwait
* Correspondence: a.niya@kcst.edu.kw

Abstract: Phishing leverages people's tendency to share personal information online. Phishing attacks often begin with an email and can be used for a variety of purposes. The cybercriminal will employ social engineering techniques to get the target to click on the link in the phishing email, which will take them to the infected website. These attacks become more complex as hackers personalize their fraud and provide convincing messages. Phishing with a malicious URL is an advanced kind of cybercrime. It might be challenging even for cautious users to spot phishing URLs. The researchers displayed different techniques to address this challenge. Machine learning models improve detection by using URLs, web page content and external features. This article presents the findings of an experimental study that attempted to enhance the performance of machine learning models to obtain improved accuracy for the two phishing datasets that are used the most commonly. Three distinct types of tuning factors are utilized, including data balancing, hyper-parameter optimization and feature selection. The experiment utilizes the eight most prevalent machine learning methods and two distinct datasets obtained from online sources, such as the UCI repository and the Mendeley repository. The result demonstrates that data balance improves accuracy marginally, whereas hyperparameter adjustment and feature selection improve accuracy significantly. The performance of machine learning algorithms is improved by combining all fine-tuned factors, outperforming existing research works. The result shows that tuning factors enhance the efficiency of machine learning algorithms. For Dataset-1, Random Forest (RF) and Gradient Boosting (XGB) achieve accuracy rates of 97.44% and 97.47%, respectively. Gradient Boosting (GB) and Extreme Gradient Boosting (XGB) achieve accuracy values of 98.27% and 98.21%, respectively, for Dataset-2.

Keywords: phishing URL; UCI and Mendeley phishing dataset; random forest; extreme gradient boosting; machine learning; SMOTE

Citation: Abdul Samad, S.R.; Balasubaramanian, S.; Al-Kaabi, A.S.; Sharma, B.; Chowdhury, S.; Mehbodniya, A.; Webber, J.L.; Bostani, A. Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection. *Electronics* **2023**, *12*, 1642. https://doi.org/10.3390/electronics12071642

Academic Editor: Cheng-Chi Lee

Received: 1 February 2023
Revised: 25 March 2023
Accepted: 27 March 2023
Published: 30 March 2023

1. Introduction

The internet and the World Wide Web are essential components of modern society. The continuous improvement of technology has made it possible to host an increasing number of beneficial web applications, which in turn have attracted many internet users. Web applications, such as social media applications, online payment systems, news, research portals, etc. become widely accessible online. In turn, this encourages the cybercriminals to launch more cyberattacks. The most common cyberattack against internet users is a phishing attack. Phishers use various social engineering methods and constantly change

their tactics to deceive people. They create phishing emails, social media postings, text messages, etc. that appear authentic to obtain sensitive information from their victims or to install malware on their computers. Fear, curiosity, urgency and greed are regularly used by phishers to convince victims to visit fraudulent websites or to click on harmful links [1]. After a successful phishing attack, the target's network is compromised; sensitive information is stolen; and the victim is left vulnerable on the internet. There were over a million phishing attacks in the first quarter of 2022, according to data provided by the anti-phishing working group (APWG). It is the most ever reported in a quarter and has continued a steady trend over the past year. APWG saw over 200,000 phishing attacks in April 2021. By March 2022, the number had almost exactly doubled, reaching 384,291 [2]. Detecting phishing web links (URLs) is becoming increasingly critical; hence, having an AI-based autonomous detection system is necessary. Researchers have proposed many different methods in order to recognize phishing Uniform Resource Locators (URLs). The most straightforward method of blocking phishing URLs is based on either a blacklist or a whitelist of URLs. The effectiveness of this detection is heavily dependent on the list of URLs. The system will either block access to the URL or decide to allow access to the URL based on whether the URL is already included in the existing list. This method needs a long list of URLs, which can be made manually or automatically, and it needs to be updated often. Otherwise, newly created phishing URLs bypass protection [3]. Next is a heuristic method that generalizes the detection strategy more effectively than blacklisting. The detecting system will function according to the rules. These rules are derived from the existing URL list. However, this needs substantial research into the many distinct methods of phishing URL generation [4,5].

To simplify developing rules based on data, researchers employ Machine learning/Deep learning models. Machine learning algorithms are data-driven computational techniques for automatically acquiring new knowledge. The algorithms improve as they get more learning samples. A branch of machine learning is called deep learning. An essential part of machine learning is selecting the features that will assist the construction of a good model that performs well on unseen data. The quality of machine-learned insights depends on the data used to train the model. Features are the building blocks of a data set and are used as independent variables in machine learning models. Researchers have carried out many studies in order to identify phishing URLs by utilizing a variety of feature sets that were taken from different datasets. Some researchers prefer to use the lexical features of phishing URLs because it is easy and safe [6]. Some are exploiting content features on websites that are extremely risky to process. Phishing websites often include malicious malware that could potentially disrupt the processing system [7,8]. Along with URL and webpage content features, other external features, such as DNS information and web page ranking, are also utilized. The visual similarity features are employed by certain studies [9]. Hybrid features, which might contain URL lexical features, webpage content features, external features and others are favored by many scholars [10]. Machine learning models, based on hybrid features, outperform other models in terms of performance. The UCI and Mendeley dataset [11] are commonly utilized for phishing URL identification due to its hybrid features. Most of the present research focuses on enhancing detection models using various machine learning methods by adding new features along with existing features in the dataset/group the features/filter or rank the features/applying ensemble the classification algorithms in different order to improve the performance. Less attention is paid to the fine-tuning factors.

To improve detection outcomes, some existing research essentially combines the proposed method with hyper-parameter optimization. Our experiment aims to combine several fine-tuning elements with the machine learning model to evaluate the performance impact without adding any additional detection methodologies, such as grouping attribute or ensemble algorithms, etc. In this experiment, we combine three different performance-improving fine-tuning factors (data balancing, hyper-parameter tuning, feature selection). Multiple fine-tuning factors help to improve the accuracy.

1. According to our investigations, data balancing results in a minor improvement in performance.
2. Usually, the tuning of hyper-parameters is iterative, consumes time for optimizing the hyper-parameter for the selected machine learning model and dataset. Based on the results of our experiments, researchers can use the different tuning parameters right away, depending on the model they choose. According to the findings, some of the models, such as SVM, KNN and GB, show a considerable improvement in accuracy when applied to Dataset-1. For Dataset-2, SVM, KNN, GNB and DT add considerable improvements.
3. The selection of features is once again an iterative process, consumes time to find an optimal number of features. In our experiment, we vary the number of features and score methods to assess performance. Results show the minimum and maximum number of features needed to achieve a higher accuracy margin and a higher accuracy, respectively.
4. Combining all these fine-tuning factors with the machine learning model gives better performance than the existing research works. To the best of our knowledge, these three tuning parameters have not so far been tested together for phishing URL detection.

The paper's contributions are as follows:

- Optimal hyper-parameters for UCI (UC Irvine) and Mendeley phishing datasets.
- Analyzing the effectiveness of eight machine learning algorithms using different performance-tuning factors, such as dataset balancing, hyper-parameter tuning and feature selection.
- Accuracy, precision, recall and F1-Score are included in the performance evaluation.
- Performance comparison.

The following summarizes the paper's structure: Section 1 outlines the problem; its significance and the various methods being employed to address it. Section 2 discusses current research studies. In Section 3, the most popular machine learning algorithms are covered. The experimental dataset and the detection approach are described in detail in Section 4. The experiments' results and contrasts of the results with prior studies are described in Section 5. The paper is concluded in Section 6.

2. Related Works

In the topic of phishing URL detection, a significant amount of research has been carried out, with the UCI and Mendeley phishing datasets. A study article by Khan et al. [12] compared the efficacy of various machine learning methods (DT, SVM, RF, NB, kNN and ANN) with three different datasets. Additionally, it examines the effectiveness of several machine learning models using a dataset with reduced dimensions. Datasets were gathered from the repository for machine learning at UCI and other online sources. The result reveals that the Random Forest method and the artificial neural network achieved 97% accuracy. Salihovic et al. [13] conducted another study that also made use of the UCI phishing dataset and the Spam dataset. For the experiment, six different machine learning algorithms (RF, kNN, ANN, SVM, LR and NB) were used. The results of the experiment show that the dataset responded differently depending on the feature selection methodology. Random Forest with Ranker + Principal Component Optimization achieved 97.33% accuracy for Phishing detection, whereas Random Forest with BestFirst + CfsSubsEval Optimization achieved 94.24% accuracy. A phisher fighter system that uses URL features and webpage features to detect phishing URLs was proposed by Vishva et al. [14]. The UCI phishing dataset was utilized for URL features, while the TF-IDF NLP approach was used to produce features for webpage content. Three machine learning algorithms (LogR, RF, and SVM) are employed in the classification of the URLs. The results show that, with a classification accuracy of 96%, the Random Forest algorithm delivers the best performance.

Hutchinson et al. [15] published an article that emphasizes the importance of the features in classification problems, especially phishing detection using a Random Forest algorithm. The experiment utilizes the UCI phishing dataset, and the features are grouped

into five categories (Sets A, B, C, D and E), which include URL based features, host-based features, ranking based features and all combined features. The result shows that Set D and E achieved higher accuracy of 95.5% and 96.5%, respectively. A comparative analysis of the effectiveness of various machine learning models using two distinct datasets, such as the UCI phishing dataset and the Mendeley phishing dataset, was presented by Sarasjati et al. [16]. Random Forest outperformed other classification techniques in the testing, with accuracy rates of 88.92% for the UCI dataset and 97.50% for the Mendeley dataset. Tama et al. [17] compared various ensembles of classifiers to detect phishing on the web. The UCI dataset was utilized. The experiment involved ensembles and single classification algorithms. Their detection performance was assessed using AUC for different resampling methods. The experiments showed that Random Forest was superior to xgboost, Rotation Forest, GBM and single classifiers C50, C-DT and CART. Karabatak [18] analyzed the efficacy of classification algorithms by comparing their results on a UCI phishing dataset. The dataset for phishing websites was reduced in dimension, and performance evaluations of classification algorithms were conducted. The results of experiments showed that minimizing the number of features improves the performance of some classification methods. Al-Sarem et al. [19] suggests a method for identifying phishing websites that uses an efficient stacking ensemble. UCI and two Mendeley Phishing datasets were utilized for the experiments. The proposed method improved detection accuracy to 97.16%, 98.58% and 97.39% for Datasets-1, -2 and -3, respectively. The existing works in the phishing URL detection are outlined in Table 1.

Table 1. Outline of related works.

Author	Year	Data Set	Result
Khan et. al. [12]	2020	UCI and other online sources	RF method and the ANN achieves 97% accuracy.
Salihovic et al. [13]	2018	UCI phishing dataset and Spam dataset	RF with Ranker + Principal Component Optimization achieves 97.33% accuracy for Phishing detection. RF with BestFirst + CfsSubsEval Optimization achieved 94.24% accuracy.
Vishva et al. [14]	2021	UCI phishing dataset, LIAR dataset	RF classifier offers the highest classification accuracy of 97%, and LR achieves 92% accuracy for URL analysis.
Hutchinson et al. [15]	2018	UCI phishing dataset	RF achieves higher accuracy for Sets D and E with 95.5% and 96.5%, respectively.
Sarasjati et al. [16]	2022	UCI phishing dataset & Mendeley phishing dataset	For the UCI dataset, RF classification reaches an accuracy level of 88.92% whereas, for the Mendeley dataset, it reaches an accuracy level of 97.50%.
Al-Sarem et al. [19]	2021	UCI phishing dataset & Mendeley phishing dataset	Optimal stacking ensemble method improved detection accuracy to 97.16%, 98.58% and 97.39% for Datasets-1, -2 and -3, respectively.

In order to solve the problem of phishing URL detection, various machine learning and deep learning models were applied. The UCI and Mendeley Phishing datasets are commonly used in much of the existing research. To increase the performance of the machine learning model, a variety of detection techniques were employed, such as adding new features to the dataset, grouping existing features into distinct categories, filtering or ranking the features and employing an ensemble classification method. Even though these approaches boost model performance, performance tuning factors are given less consideration. Our study focuses on combining multiple performance tuning factors with a machine learning model, improving the model's performance.

Purpose of the research work: The purpose of our research is to increase the accuracy of phishing detection by integrating multiple performance tuning factors with a machine learning model and evaluating the performance without the inclusion of additional detection methodologies.

Research Gap: Most of the present research focuses on enhancing detection models using various machine learning methods by adding new features along with existing features in the dataset/group the features/filter or rank the features/applying ensemble the classification algorithms in different order to improve the performance. Less attention is paid to the fine-tuning factors. Some of the existing research uses merely hyper-parameter tuning combined with the approach they propose to get better detection results. Our experiment aims to combine multiple fine-tuning elements with the machine learning model to evaluate the performance impact without adding any additional detection methodologies, such as grouping attribute, ensemble algorithms, etc.

Our research evaluates the effectiveness of the most used machine learning model and focuses on three different fine-tuning parameters. The model's performance improves with each fine-tuning factor. Combining all the three performance-tweaking factors with a machine learning model significantly enhances the performance.

3. Background

Detecting phishing URLs is a binary classification problem. Binary classification in machine learning divides raw data into two groups. The most common classification algorithms for phishing URL detection are shown in Table 2. In our experimental study, these algorithms are used with tuning factors to assess performance.

Table 2. Classification algorithms.

Algorithm	Description
Logistic Regression (LogR)	The most effective supervised learning approach for binary classification is logistic regression since its prediction is discrete. The sigmoid function is applied in logistic regression to convert any real number to a value between 0 and 1 [20].
Support Vector Machine (SVM)	SVM can classify and predict. SVM draws a border to differentiate classes. The process for determining the decision boundary is the most critical part of SVM algorithms. Each data is shown in n-dimensions prior to the decision boundary being built. When creating the decision boundary, care is made to maintain the greatest practical distance from the support vectors [21].
Decision Tree (DT)	DT is a classification and regression algorithm. Each internal node in a decision tree, which has a tree structure equivalent to a flowchart, denotes a feature. Each branch in the tree represents a decision rule, and each leaf node in the tree denotes the result [22].
K Nearest Neighbors (KNN)	KNN is a classification algorithm. The classification of a new data point is based on its similarity to a certain group of nearby data points, making it one of the most straightforward and popular classification techniques [23].
Random Forest (RF)	Random Forest is a decision-tree-based classifier. It is made up of many different decision trees. Random Forest accumulates classifications and uses the most voted prediction to classify a new occurrence. A subset of the original dataset is fed into each tree as input. Furthermore, a subset of optional features is chosen at random to develop the tree at each node [24].
Naïve Bayes (NB)	The NB classification method, which relies on the Bayes Theorem, assumes that all predictor features are independent. It calculates each class's probability and picks the highest one [25].
Gradient Boosting (GB)	Gradient Boosting is an algorithm for classification and regression in machine learning. It is an ensemble learner, which means that it will merge many independent models to create a final model. The individual prediction capabilities of these models are subpar, and they are prone to overfitting. However, combining many of these subpar models into an ensemble will provide a result that is noticeably superior overall. Thus, it reduces the prediction error. Gradient boosting employs decision trees as a weak learner [26].
Extreme Gradient Boosting (XGB)	To maximize efficiency, adaptability and portability, the XGBoost distributed gradient boosting library was developed. The machine learning techniques are implemented using the Gradient Boosting framework. It solves various data science issues quickly and accurately via parallel tree boosting [26].

4. Methodology

The main objective of our experimental study is to evaluate the performance of the fine-tuned machine learning model. The experiment makes use of the eight most common machine learning algorithms listed in Table 2. We employ three different types of tuning factors, such as dataset balancing, hyper-parameter optimization and feature selection. In each case, the performance of the algorithm with a specific dataset is presented in Section 5. Figure 1 depicts an overview of the system process flow.

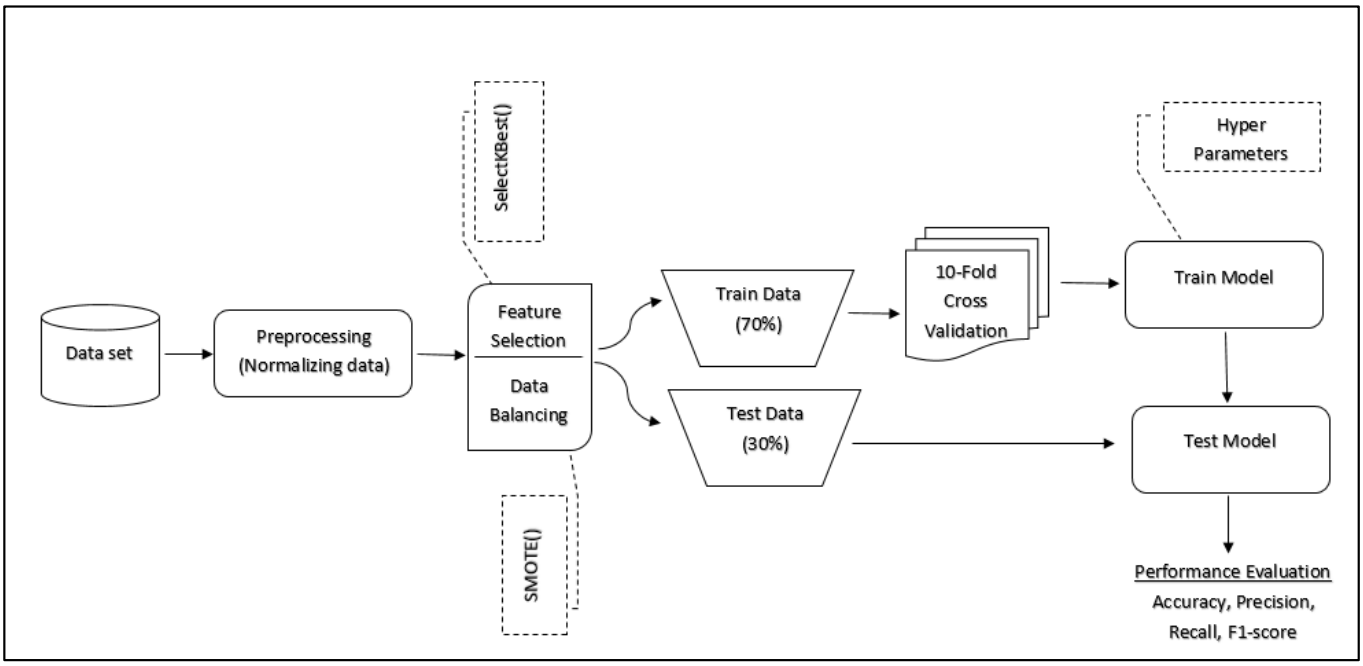

Figure 1. Process flow.

The most popular datasets from UCI and Mendeley Repository are used in the experiment. Before running the experiment, the dataset was normalized, balanced (if needed) and divided into training and test sets. We apply the selectKbest filtering method, which employs a variety of scoring functions, including correlations and mutual information gain, to choose a subset of features from the full set in order to reduce the data dimension. The experiment uses all eight machine learning models that are listed in Table 1. Four metrics, including accuracy, precision, recall and F1-score, are used to assess each machine learning model's performance [6]. These metrics are obtained from the confusion matrices, as given in Table 3. In addition, it is essential to evaluate the stability of every machine learning algorithm. Cross-validation is a method for evaluating the effectiveness of a model by using a subset of the input data as training and a portion of the input data as testing that has never been used before. Our experiment used the 10-fold cross validation method [27] to guarantee the machine learning algorithm's stable performance.

Table 3. Confusion matrix.

		Predicted	
		Positive (1) (Malicious)	Negative (0) (Benign)
Actual	Positive (1) (Malicious)	TP	FN
	Negative (0) (Benign)	FP	TN

T = True, F = False, P = Positive, N = Negative.

Accuracy (acc)

Shows how many accurate predictions were made compared to the total number of predictions.

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision (Pr)

Precision is a way to measure how many positive class predictions are actually true.

$$Pr = \frac{TP}{TP + FP}$$

Recall (Rc)

Recall measures how many instances of a given class were correctly predicted out of all instances of that class in the dataset.

$$Rc = \frac{TP}{TP + FN}$$

F1-Score (FS)

F-Measure combines precision and recall into one score.

$$FS = \frac{2\ x\ Pr\ x\ Rc}{Pr + Rc}$$

TP = Truly Positive (TP), TN = Truly Negative, FP =False Positive and FN = False Negative.

Experiments Dataset

The initial dataset utilized in this experiment was obtained from the UCI Repository [28]. This dataset has 11,055 records and 31 features. Out of the 31 features, 30 of which are independent features and 1 of which is a target, which is categorized into 4 groups, such as address bar-based features, abnormality-based features, HTML and JavaScript-based features, and domain-based features. In the 11,055 instances, 6157 reliable websites (1) and 4898 phishing websites (-1) make up the dataset.

The second dataset, which was obtained from the Mendeley repository [29], was made up of 48 features that were taken from a total of 10,000 online pages, 5000 of which were phishing sites and 5000 of which were legitimate websites. Alexa and Common Crawl are used to compile lists of legitimate websites; whereas Phish-Tank and OpenPhish are used to compile lists of malicious ones. Binary labels, such as 0 for legitimate and 1 for phishing, are present in this dataset. These features are divided into three groups: features based on anomalies, features based on HTML/JavaScript and features based on the address bar. Examples of address bar-based functionality include the port number and length of a website's URL. HTML and JavaScript techniques are incorporated within the website's source code for both abnormal-based and HTML/JavaScript features. The downloading of items from other websites is an illustration of an abnormal-based functionality [30]. The dataset's overview is presented in Table 4. Tables 5 and 6 provide the dataset's characteristics.

Table 4. Summary of the datasets.

	Data Source	Number of Instances and Features	Types of Features	Number of Legitimate & Phishing URLs	Target Class
Dataset-1	UCI Repository [28]	11,055 instances and 31 features	Address bar-based features, abnormality-based features, HTML and JavaScript-based features, domain-based features	6157 Legitimate URLs. (56%) 4898 phishing URLs. (44%)	1- Legitimate −1- Phishing
Dataset-2	Mendeley Repository [29]	10,000 instances and 49 features	Address bar-based, Abnormal-based and HTML/JavaScript-based features	5000 Legitimate URLs. 5000 phishing URLs.	0- Legitimate 1- Phishing

Table 5. Features of Dataset-1.

No.	Features	Possible Values	No.	Features	Possible Values
1	having_IP_Address	{1, −1}	16	SFH	{0, 1, −1}
2	URL_Length	{0, 1, −1}	17	Submitting_to_email	{1, −1}
3	Shortening_Service	{1, −1}	18	Abnormal_URL	{1, −1}
4	having_At_Symbol	{1, −1}	19	Redirect	{0, 1}
5	double_slash_redirecting	{1, −1}	20	on_mouseover	{1, −1}
6	Prefix_Suffix	{1, −1}	21	RightClick	{1, −1}
7	having_Sub_Domain	{0, 1, −1}	22	popUpWindow	{1, −1}
8	SSLfinal_State	{0, 1, −1}	23	Iframe	{1, −1}
9	Domain_registration_length	{1, −1}	24	age_of_domain	{1, −1}
10	Favicon	{1, −1}	25	DNSRecord	{1, −1}
11	port	{1, −1}	26	web_traffic	{0, 1, −1}
12	HTTPS_token	{1, −1}	27	Page_Rank	{1, −1}
13	Request_URL	{1, −1}	28	Google_Index	{1, −1}
14	URL_of_Anchor	{0, 1, −1}	29	Links_pointing_to_page	{0, 1, −1}
15	Links_in_tags	{0, 1, −1}	30	Statistical_report	{1, −1}
			31	Result	{1, −1}

Table 6. Features of Dataset-2.

No.	Features	Min-Max Values	No.	Features	Min-Max Values
1	NumDots	[1, 21]	25	NumSensitiveWords	[0, 3]
2	SubdomainLevel	[0, 14]	26	EmbeddedBrandName	[0, 1]
3	PathLevel	[0, 18]	27	PctExtHyperlinks	[0.0, 1.0]
4	UrlLength	[12, 253]	28	PctExtResourceUrls	[0.0, 1.0]
5	NumDash	[0, 55]	29	ExtFavicon	[0, 1]
6	NumDashInHostname	[0, 9]	30	InsecureForms	[0, 1]
7	AtSymbol	[0, 1]	31	RelativeFormAction	[0, 1]

Table 6. *Cont.*

No.	Features	Min-Max Values	No.	Features	Min-Max Values
8	TildeSymbol	[0, 1]	32	ExtFormAction	[0, 1]
9	NumUnderscore	[0, 18]	33	AbnormalFormAction	[0, 1]
10	NumPercent	[0, 19]	34	PctNullSelfRedirectHyperlinks	[0.0, 1.0]
11	NumQueryComponents	[0, 23]	35	FrequentDomainNameMismatch	[0, 1]
12	NumAmpersand	[0, 22]	36	FakeLinkInStatusBar	[0, 1]
13	NumHash	[0, 1]	37	RightClickDisabled	[0, 1]
14	NumNumericChars	[0, 111]	38	PopUpWindow	[0, 1]
15	NoHttps	[0, 1]	39	SubmitInfoToEmail	[0, 1]
16	RandomString	[0, 1]	40	IframeOrFrame	[0, 1]
17	IpAddress	[0, 1]	41	MissingTitle	[0, 1]
18	DomainInSubdomains	[0, 1]	42	ImagesOnlyInForm	[0, 1]
19	DomainInPaths	[0, 1]	43	SubdomainLevelRT	[−1, 1]
20	HttpsInHostname	[0, 0]	44	UrlLengthRT	[−1, 1]
21	HostnameLength	[4, 137]	45	PctExtResourceUrlsRT	[−1, 1]
22	PathLength	[0, 161]	46	AbnormalExtFormActionR	[−1, 1]
23	QueryLength	[0, 188]	47	ExtMetaScriptLinkRT	[−1, 1]
24	DoubleSlashInPath	[0, 1]	48	PctExtNullSelfRedirectHyperlinksRT	[−1, 1]
			49	CLASS_LABEL	[0, 1]

5. Experiment Results

The performance effects of three different fine-tuning factors on eight different classification algorithms are examined in this section. Section 5.1 investigates the effects of data balancing on classification algorithms. Section 5.2 assesses the performance improvement by using hyper-parameters. Section 5.3 analyzes the ideal feature set to achieve the best performance.

5.1. Investigates the Effects of Data Balancing on Classification Algorithms

Balanced data has an even distribution of the target class, but unbalanced data can be different. The imbalanced dataset could lead to skewed results. In other words, unbalanced data can lead to results that are skewed in favor of the majority class. The machine learning algorithm is not able to learn from the minority class. Subsequently, whatever test data is given to the machine learning algorithm will be assigned to the majority class. Model performance is often measured by prediction accuracy. In the case of an imbalanced dataset, this is inappropriate [31]. Zheng et al. [32] examined the effects of imbalanced data on machine learning models by evaluating eight well-known machine learning models on 48 different imbalanced datasets. The results demonstrate that, as the imbalance rate increases, the classification accuracy of the algorithms decreases.

To solve this problem, various methods might be used. Common and straightforward are undersampling and oversampling. Uneven datasets can be balanced using the technique of undersampling. In this approach, the size of the class that belongs to the majority is reduced while all the data from the minority class is retained, while oversampling will raise the quantity of minority data samples to match the majority. The minority class's data samples will be duplicated. Usually, oversampling is better than undersampling because deleting data can mean losing critical features. For the minority class, synthetic samples are created using the synthetic minority oversampling technique (SMOTE) [33,34]. A symbolic representation of oversampling using SMOTE is depicted in Figure 2, and the

algorithm is presented in Algorithm 1. This approach overcomes random oversampling's overfitting problem. Dataset-1 is having 56% legitimate URLs and 44% phishing URLs as shown in Table 7. So, we apply SMOTE oversampling technique to balance the Dataset-1 for our experiment. With the balanced and unbalanced dataset, the experiment displays the performance of eight different machine learning algorithms. The results are shown in Table 8.

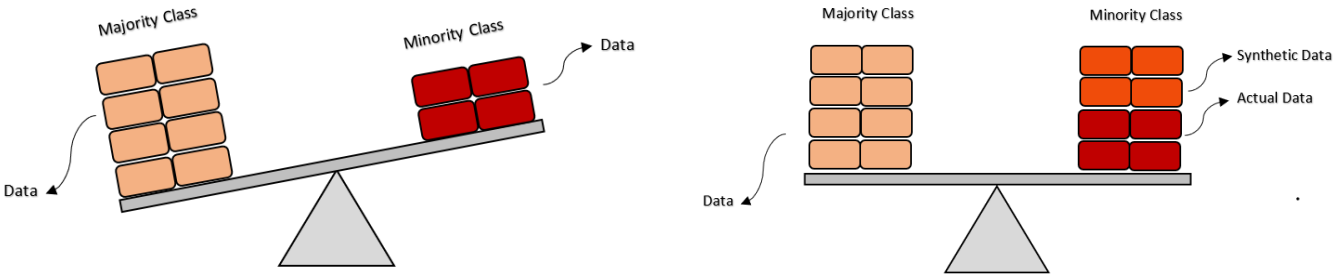

Figure 2. Oversampling using SMOTE.

Algorithm 1: SMOTE

Input: Imbalanced URL (Phishing and Benign URLs) Dataset D
Output: Balanced URL (Phishing and Benign URLs) Dataset D'
Load (URL Dataset)
Find the Minority Class Set (MIN_C)
Take the random sample in MIN_C ($M \in MIN\_C$)
For each X in M:
  Find the K -Nearest Neighbor (N) of X.
  For each X_k in N:
    Identify the vector between X and X_k .
      Multiply the vector by a random number between 0 and 1 and add to the current data point.

$$X' = X + rand(0,1) * |X - X_k|$$

D' = Add X' to D
return D'

Table 7. Experiment dataset for Data Balancing.

Dataset	Actual Number of Records in the Dataset	Number of Records after Balancing the Dataset
Dataset-1 (UCI Repository)	Legitimate URLs: 6157 (Majority Class)	Legitimate URLs: 6157
	Phishing URLs: 4898 (Minority Class)	Phishing URLs: 6157

Table 8. Performance comparison of balanced dataset with imbalanced dataset (Dataset-1).

Classification Algorithm	Type of Dataset	Accuracy%	Precision%	Recall%	F1-Score
Logistic Regression (LogR)	Imbalanced	92.537	92.382	94.397	93.375
	Balanced	92.261	91.481	93.211	92.335
Support Vector Machine (SVM)	Imbalanced	94.744	93.986	96.768	95.354
	Balanced	94.949	94.046	95.988	95.004
Bernoulli Naïve Bayes (BNB)	Imbalanced	90.285	91.170	91.473	91.305
	Balanced	90.596	90.863	90.352	90.590

Table 8. *Cont.*

Classification Algorithm	Type of Dataset	Accuracy%	Precision%	Recall%	F1-Score
K-Neighbors Classifier (KNN)	Imbalanced	94.708	94.689	95.891	95.283
	Balanced	94.924	95.204	94.640	94.909
Decision Tree (DT)	Imbalanced	96.182	96.378	96.800	96.579
	Balanced	96.694	96.699	96.702	96.691
Random Forest (RF)	Imbalanced	97.223	96.816	98.262	97.529
	Balanced	97.425	97.172	97.710	97.435
Gradient Boosting (GB)	Imbalanced	94.699	94.501	96.086	95.283
	Balanced	94.713	94.323	95.176	94.742
Extreme Gradient Boosting (XGB)	Imbalanced	97.286	96.980	98.197	97.582
	Balanced	97.028	96.903	97.174	97.033

According to the experiment results in Table 8 and Figure 3, minor performance improvement occurs after balancing the Dataset-1. The algorithms SVM, BNB, KNN, DT, RF and GB provide marginal improvements in performance. The RF and XGB algorithms achieve the highest accuracy (%) for the balanced dataset with scores of 97.425 and 97.028, respectively, and the XGB and RF algorithms achieve the highest F1-Score with scores of 97.582 and 97.529, respectively, for the imbalanced dataset.

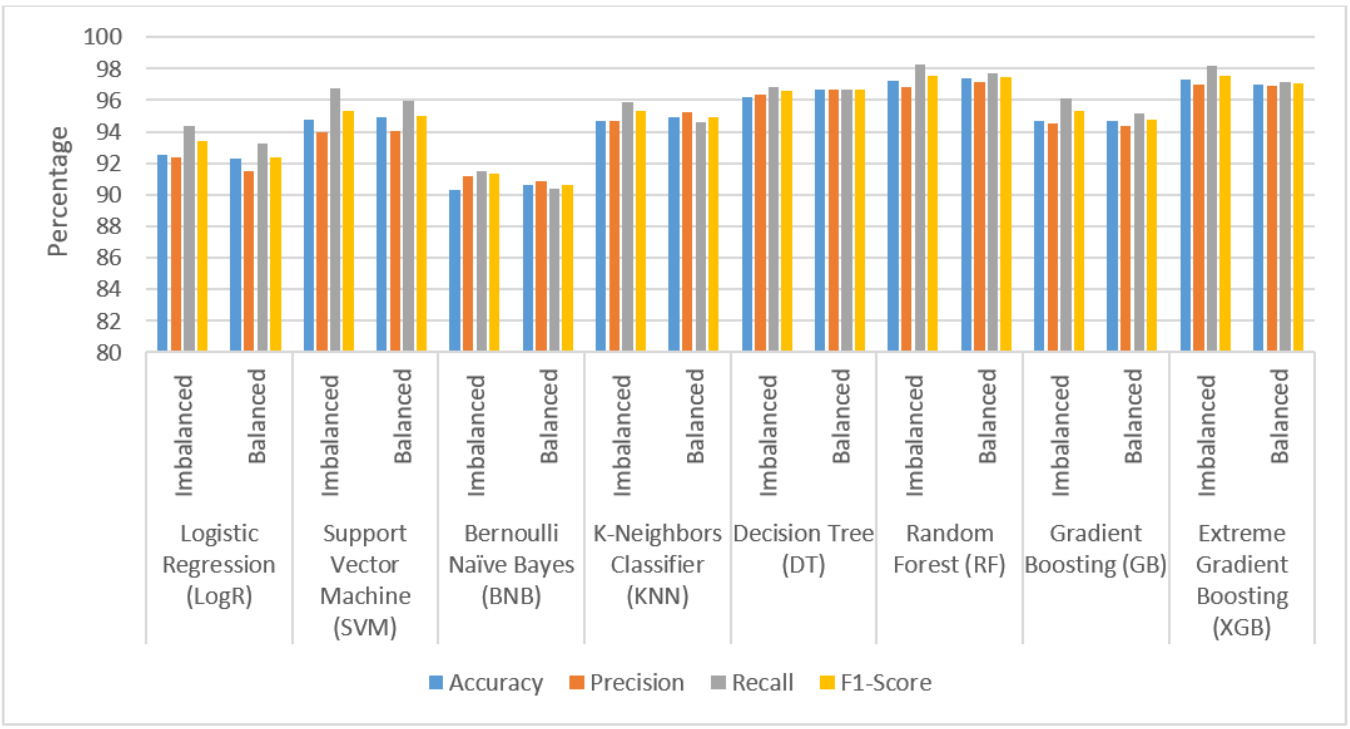

Figure 3. Performance comparison of balanced dataset with imbalanced dataset (Dataset-1).

5.2. Examine the Performance Impact of Hyper-Parameter Tuning on Classification Algorithms

Hyper-parameters are the parameters that are set by the user to control the learning process of the machine learning model. Estimated model parameters that failed to minimize the loss function are a byproduct of poorly tuned hyper-parameters. This implies that the model is more inaccurate. The optimal values for the hyper-parameters can be calculated either manually or automatically. When manually tuning hyper-parameters, it is often

best to start with the recommended values or rules of thumb and then to search through a range of values by trial and error. However, manual tuning requires a lot of effort and takes a lot of time. It is impractical if there are several hyper-parameters with a wide range. Automated methods of hyper-parameter tuning use an algorithm to identify the optimum values. Grid search, Random search and Bayesian optimization are common automated methods. In our experiment, the grid search approach was employed. Grid search is a "brute force" technique for hyper-parameter tweaking. The model is first fit using a grid of all possible values for each discrete hyper-parameter. Every set's performance from the model is recorded, and the set with the best performance is chosen.

The experimental datasets are listed in Table 9. Hyper-parameters and values for eight different classification models are provided in Table 10. Table 11 and Figure 4 show the performance comparison of the hyper-parameter tuned and untuned model for Dataset-1. Table 12 and Figure 5 show the performance comparison of hyper-parameter tuned and untuned model for Dataset-2.

Table 9. Experiment dataset for Hyper-Parameter Tuning.

Dataset	Number of Records after Balancing the Dataset
Dataset-1 (UCI Repository)	Legitimate URLs: 6157
	Phishing URLs: 6157
Dataset-2 (Mendeley Repository)	Legitimate URLs: 5000
	Phishing URLs: 5000

Table 10. Hyper-parameters for classification algorithms.

Classification Algorithm	Hyper-parameters	Data Set −1 Values	Data Set −2 Values
Logistic Regression (LogR)	C	10	100
	penalty	l2	l1
	solver	lbfgs	liblinear
	max_iter	–	100
Support Vector Machine (SVM)	C	10.0	10.0
	gamma	0.1	0.1
	kernel	rbf	rbf
Gaussian Naïve Bayes (GNB)	var_smoothing	–	1.2328467394420658e−05
Bernoulli Naïve Bayes (BNB)	alpha	0.0	–
	binarize	0.0	–
	class_prior	None	–
	fit_prior	True	–
K-Neighbors Classifier (KNN)	algorithm	auto	auto
	metric	manhattan	manhattan
	n_neighbors	7	3
	weights	distance	distance
Decision Tree (DT)	criterion	entropy	entropy
	max_depth	None	40
	max_features	sqrt	log2
	splitter	best	best

Table 10. *Cont.*

Classification Algorithm	Hyper-parameters	Data Set −1 Values	Data Set −2 Values
Random Forest (RF)	max_features	log2	log2
	n_estimators	100	200
	max_depth	40	None
	criterion	gini	entropy
	min_samples_leaf	–	1
	min_samples_split	–	2
Stochastic Gradient Boosting (SGB)	learning_rate	1	0.1
	max_features	log2	log2
	n_estimators	1000	1000
	criterion		squared_error
Extreme Gradient Boosting (XGB)	colsample_bylevel _	0.8	0.5
	colsample_bytree	–	0.5
	max_depth	–	30
	learning_rate	–	0.2
	n_estimators	–	1200
	reg_alpha	0.5	–
	reg_lambda	0.5	–

Table 11. Performance comparison of hyper-parameter tuned and untuned model for Dataset-1.

Classification Algorithm	Type of Machine Learning Model	Accuracy%	Precision%	Recall%	F1-Score%
Logistic Regression (LogR)	No hyper-parameter	92.301	91.491	93.292	92.379
	Tuned with hyper- parameters	92.374	91.558	93.373	92.453
Support Vector Machine (SVM)	No hyper-parameter	94.949	94.046	95.988	95.004
	Tuned with hyper- parameters	96.898	96.315	97.547	96.921
Bernoulli Naïve Bayes (BNB)	No hyper-parameter	90.596	90.863	90.352	90.590
	Tuned with hyper- parameters	90.531	90.62	90.515	90.549
K-Neighbors Classifier (KNN)	No hyper-parameter	94.924	95.204	94.640	94.909
	Tuned with hyper- parameters	96.832	96.541	97.19	96.85
Decision Tree (DT)	No hyper-parameter	96.694	96.699	96.702	96.691
	Tuned with hyper- parameters	96.475	96.483	96.475	96.473
Random Forest (RF)	No hyper-parameter	97.425	97.172	97.710	97.435
	Tuned with hyper- parameters	97.466	97.163	97.807	97.477
Gradient Boosting (GB)	No hyper-parameter	94.713	94.323	95.176	94.742
	Tuned with hyper- parameters	97.076	96.657	97.547	97.092
Extreme Gradient Boosting (XGB)	No hyper-parameter	97.028	96.903	97.174	97.033
	Tuned with hyper- parameters	97.182	96.93	97.482	97.195

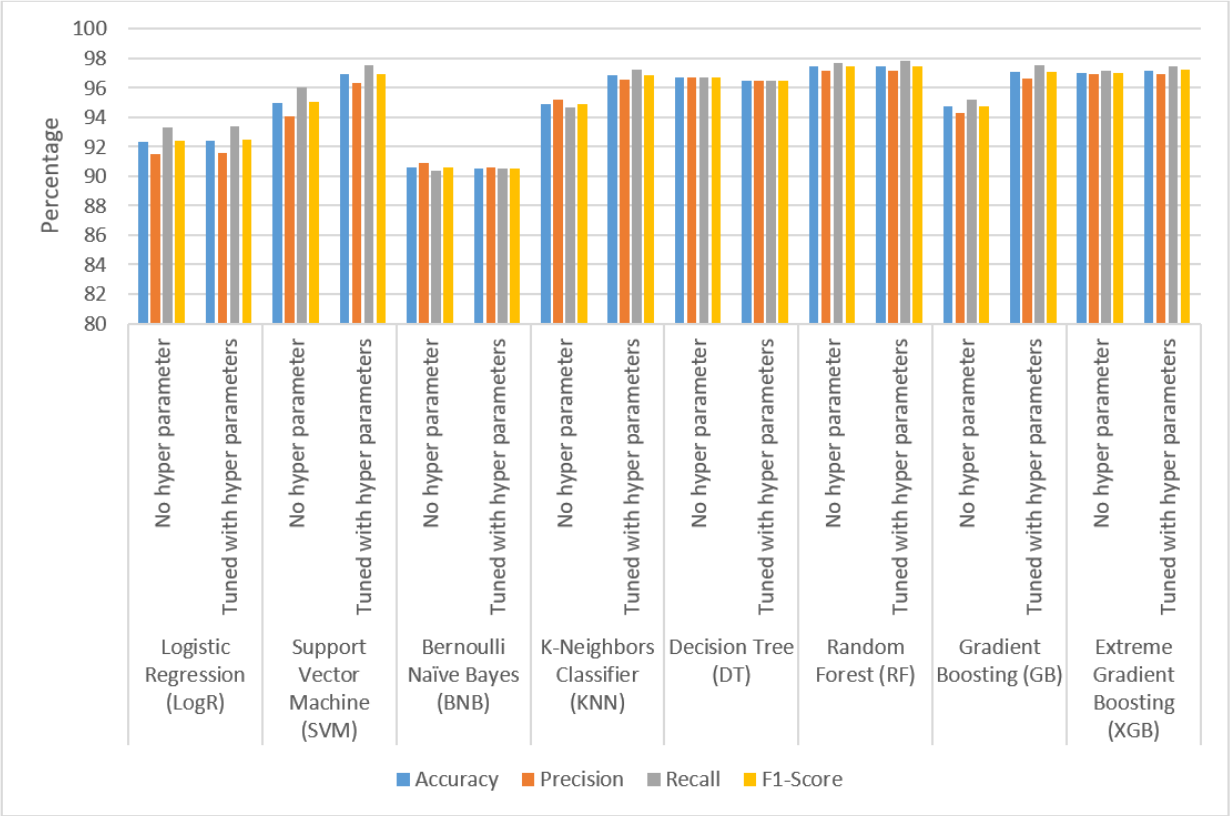

Figure 4. Performance comparison of hyper-parameter tuned and untuned model for Dataset-1.

Table 12. Performance comparison of hyper-parameter tuned and untuned model (Dataset-2).

Classification Algorithm	Type of Machine Learning Model	Accuracy%	Precision%	Recall%	F1-Score%
Logistic Regression (LogR)	No hyper-parameter	93.01	92.046	94.22	93.102
	Tuned with hyper- parameters	93.82	93.052	94.76	93.886
Support Vector Machine (SVM)	No hyper-parameter	94.48	94.48	94.48	94.48
	Tuned with hyper- parameters	96.66	96.66	96.66	96.66
Gaussian Naïve Bayes (GNB)	No hyper-parameter	83.73	93.94	72.1	81.441
	Tuned with hyper- parameters	84.4	92.878	74.5	82.621
K-Neighbors Classifier (KNN)	No hyper-parameter	93.63	94.415	92.76	93.571
	Tuned with hyper- parameters	94.73	94.638	94.86	94.742
Decision Tree (DT)	No hyper-parameter	95.57	95.593	95.62	95.581
	Tuned with hyper- parameters	94.52	94.721	94.3	94.503
Random Forest (RF)	No hyper-parameter	97.62	98.072	97.16	97.61
	Tuned with hyper- parameters	97.74	98.212	97.26	97.73
Gradient Boosting (GB)	No hyper-parameter	97.25	97.261	97.24	97.248
	Tuned with hyper- parameters	97.98	98.01	97.96	97.98
Extreme Gradient Boosting (XGB)	No hyper-parameter	98.15	98.461	97.84	98.145
	Tuned with hyper- parameters	98.26	98.537	97.98	98.253

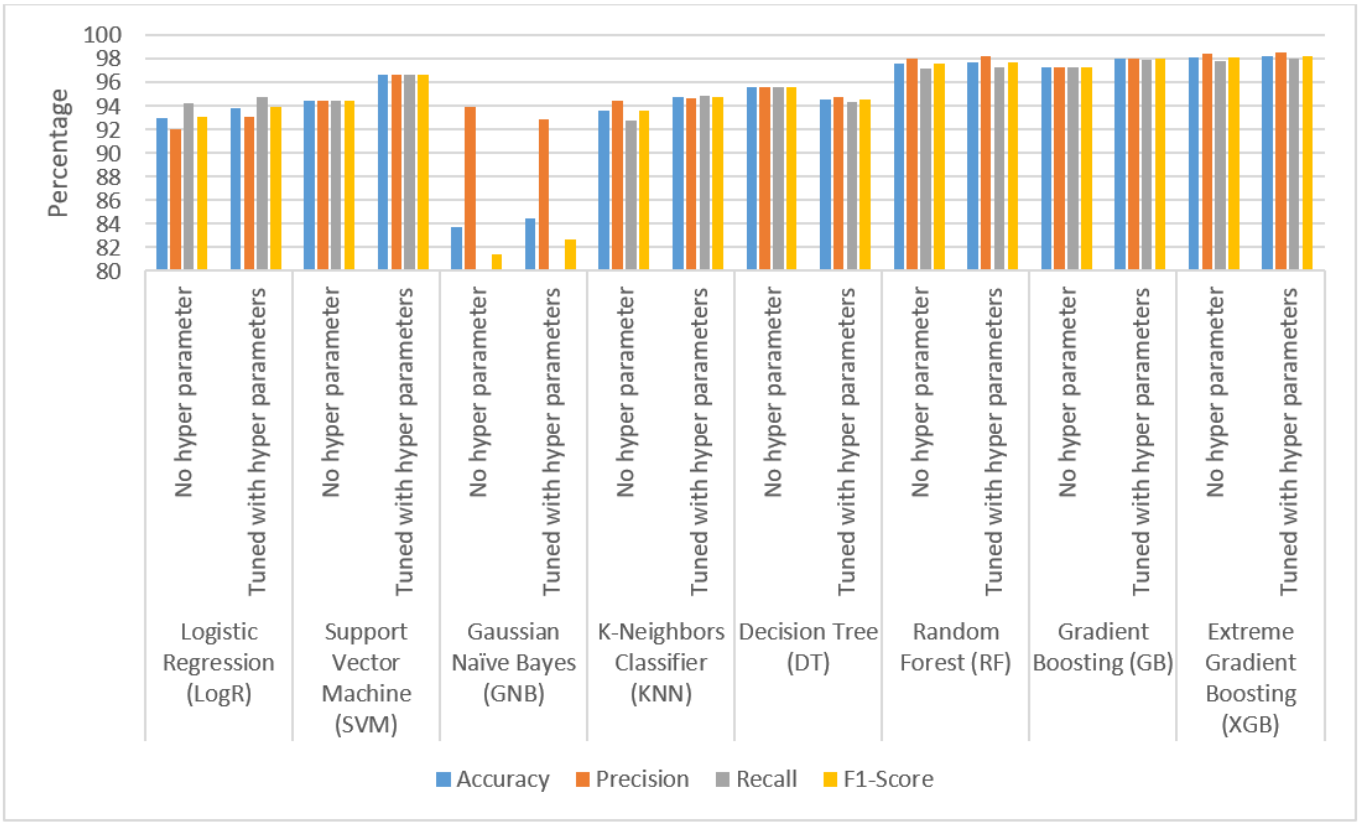

Figure 5. Performance comparison of hyper-parameter tuned and untuned model (Dataset-2).

As stated in Tables 11 and 12, most common classification algorithms exhibit modest to significant changes in accuracy (%) value for Dataset-1 and Dataset-2, as illustrated in Figures 4 and 5. GB, XGB and RF achieve the highest performance for Dataset-1 with 97.076, 97.182 and 97.466 accuracies, respectively. RF, GB and XGB achieve the highest performance for Dataset-2, with 97.74, 97.98 and 98.26 accuracies, respectively.

5.3. Examine the Minimum and Maximum Number of Features Needed for Optimal Performance

The process of choosing a subset of variables from the original data set to utilize as inputs in a machine learning model is known as feature selection. Typically, data collection includes a sizable number of attributes. A variety of methods can be used to determine which of these features are useful in generating predictions. Models for machine learning with fewer features have benefits such as less redundancy, simpler understanding, faster training and simpler deployment. Simpler models might be more generalizable and exhibit less overfitting. A model with too many variables frequently becomes noisy. By removing noisy and useless information, the model may be better able to adapt to fresh, unforeseen data, resulting in predictions that are more accurate. A feature selection procedure combines a search technique and an evaluation method. The search technique suggests new subsets of features, and the evaluation metric assesses the subset's quality. These methods can be classified as a filter, wrapper or embedding. Our experiment employs the filtering method. The first phase of a filter algorithm is to rank features according to some criteria, and the second is to select the features with the highest ranking to use in training the machine learning models. The experiment makes use of SelectKBest in order to choose the features that have the best variance. There are two inputs to this method. One is the scoring metric, which is the ANOVA F-value between label and feature for classification tasks or mutual_info_classif for mutual information for a discrete target or chi2 for chi-squared stats of non-negative features for classification tasks, and another is the value of K, which denotes the number of features in the final dataset. Table 13 shows the features of two

separate datasets utilized in the experiment. The mutual information gains score for every feature in Dataset-1, and Dataset-2 is depicted in Figures 6 and 7 respectively. Data were normalized and balanced prior to the experiment. Hyper-parameter tuning is also used to produce better results with fewer features. The K parameter for the selectKbest() method ranges from 2 to the total number of features in the dataset. Tables 14 and 15 display the experiment's outcomes.

Table 13. Experiment dataset for feature selection.

Dataset	Number of Records	Number of Features
Dataset-1 (UCI Repository)	Legitimate URLs: 6157 Phishing URLs: 6157 (after SMOTE)	30
Dataset-2 (Mendeley Repository)	Legitimate URLs: 5000 Phishing URLs: 5000	49

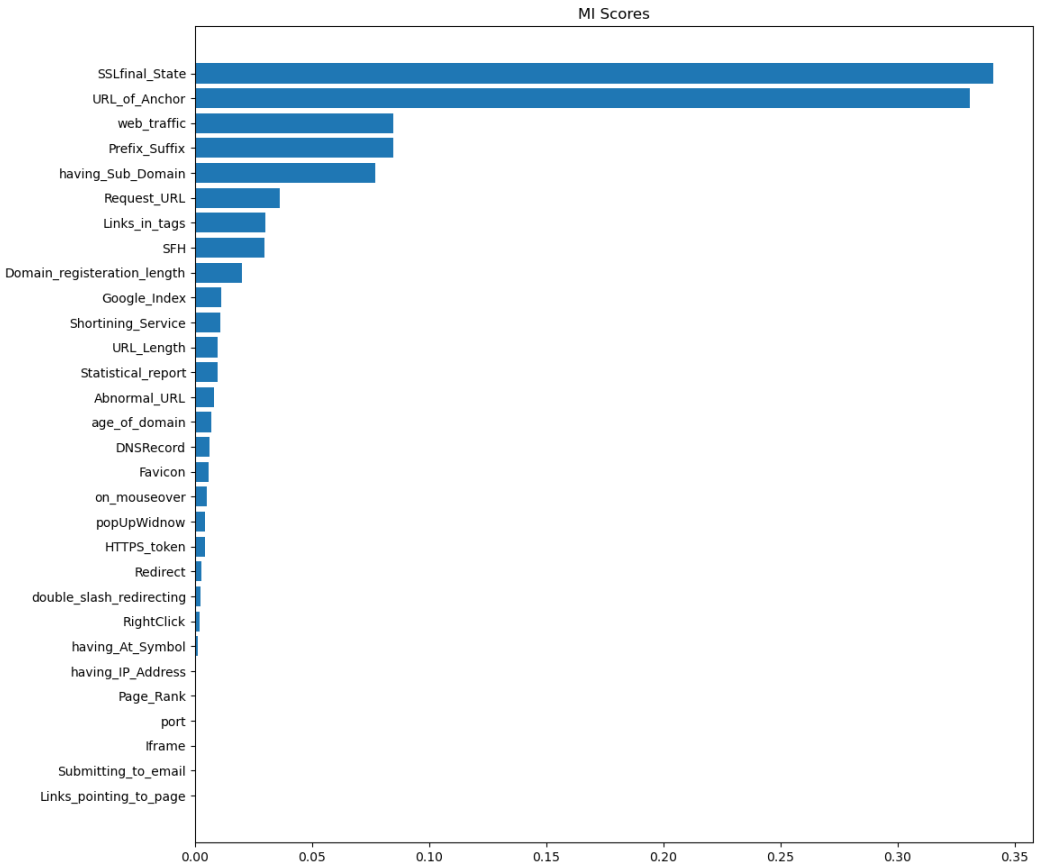

Figure 6. Mutual information gain score for Dataset-1.

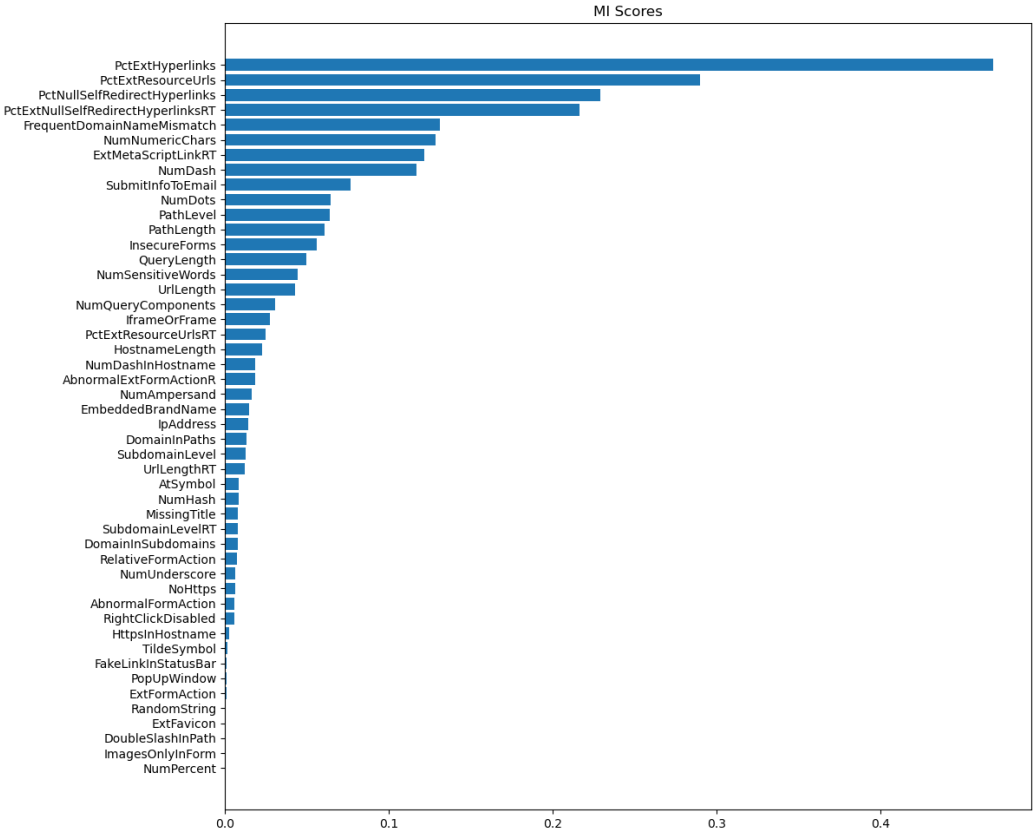

Figure 7. Mutual information gain score for Dataset-2.

Table 14. Maximum and minimum number of features for optimal performance for Dataset-1.

Classification Algorithm	Scoring Function	Best Accuracy%	Maximum No.of Features to Reach Highest Accuracy	Minimum No.of Features to Reach High Accuracy Margin
Logistic Regression (LogR)	f_classif–ANOVA F	92.33	28	10 (Accuracy %: 92.06)
		92.29	25	
		92.25	26	
	mutual_info_classif–Mutual information	92.33	29	9 (Accuracy %:92.03)
		92.27	28	
		92.24	30	
Support Vector Machine (SVM)	f_classif–ANOVA F	97.0	29	20 (Accuracy %:96.02)
		96.99	30	
		96.97	28	
	mutual_info_classif–Mutual information	97.04	28	19 (Accuracy %: 96.12)
		96.90	27	
		96.87	26	

Table 14. *Cont.*

Classification Algorithm	Scoring Function	Best Accuracy%	Maximum No.of Features to Reach Highest Accuracy	Minimum No.of Features to Reach High Accuracy Margin
Bernoulli Naïve Bayes (BNB)	f_classif–ANOVA F	91.27	8	4 (Accuracy %:90.06)
		90.96	6	
		90.75	17	
	mutual_info_classif–Mutual information	91.25	8	5 (Accuracy %:91.22)
		91.22	5	
		90.86	6	
K-Neighbors Classifier (KNN)	f_classif–ANOVA F	96.91	30	18 (Accuracy %:96.02)
		96.84	27	
		96.82	29	
	mutual_info_classif–Mutual information	97.04	26	17 (Accuracy %:96.04)
		96.88	23	
		96.87	27	
Decision Tree (DT)	f_classif–ANOVA F	96.68	25	20 (Accuracy %:96.18)
		96.58	26	
		96.53	27	
	mutual_info_classif–Mutual information	96.41	27	18 (Accuracy %:96.20)
		96.39	23	
		96.32	24	
Random Forest (RF)	f_classif–ANOVA F	97.44	30	21 (Accuracy %:97.13)
		97.40	25	
		97.36	27	
	mutual_info_classif–Mutual information	97.41	28	22 (Accuracy %:97.19)
		97.40	30	
		97.39	29	
Gradient Boosting (GB)	f_classif–ANOVA F	97.47	29	22 (Accuracy %:97.18)
		97.43	27	
		97.41	26	
	mutual_info_classif–Mutual information	97.35	29	26 (Accuracy %: 97:05)
		97.29	28	
		97.20	27	
Extreme Gradient Boosting (XGB)	f_classif–ANOVA F	97.26	27	20 (Accuracy %:97.04)
		97.17	29	
		97.16	25	
	mutual_info_classif–Mutual information	97.19	29	22 (Accuracy %: 97.08)
		97.09	30	
		97.08	22	

Table 15. Maximum and minimum number of features for optimal performance for Dataset-2.

Classification Algorithm	Scoring Function	Best Accuracy%	Maximum No. of Features to Reach Highest Accuracy	Minimum No. of Features to Reach Accuracy Margin
Logistic Regression (LogR)	f_classif–ANOVA F	93.87	44	36 (Accuracy %: 93.27)
		93.86	43	
		93.84	46	
	mutual_info_classif–Mutual information	93.85	45	31 (Accuracy %: 93.04)
		93.82	42	
		93.77	38	
	chi2–Chi-squared stats	93.88	43	28 (Accuracy %: 93.12)
		93.85	45	
		93.84	46	
Support Vector Machine (SVM)	f_classif–ANOVA F	96.66	47	31 (Accuracy %: 96.06)
		96.64	43	
		96.63	44	
	mutual_info_classif–Mutual information	96.79	42	25 (Accuracy %: 96.29)
		96.78	44	
		96.72	45	
	f_classif–ANOVA F	96.79	42	28 (Accuracy %: 96.05)
		96.78	43	
		96.77	44	
Gaussian Naïve Bayes (GNB)	f_classif–ANOVA F	84.5	45	21 (Accuracy %: 84.07)
		84.4	46	
		83.94	42	
	mutual_info_classif–Mutual information	86.99	36	25 (Accuracy %:85.01)
		85.69	28	
		85.08	27	
	chi2–Chi-squared stats	84.52	38	21 (Accuracy %:84.09)
		84.46	35	
		84.45	34	
K-Neighbors Classifier (KNN)	f_classif–ANOVA F	94.83	44	14 (Accuracy %:94.2)
		94.73	47	
		94.64	43	
	mutual_info_classif–Mutual information	95.41	41	13 (Accuracy %:95.31)
		95.33	24	
		95.31	13	
	chi2–Chi-squared stats	95.13	11	11 (Accuracy %:95.13)
		94.98	13	
		94.83	14	

Table 15. *Cont.*

Classification Algorithm	Scoring Function	Best Accuracy%	Maximum No. of Features to Reach Highest Accuracy	Minimum No. of Features to Reach Accuracy Margin
Decision Tree (DT)	f_classif–ANOVA F	95.53	12	9 (Accuracy:95.17)
		95.35	19	
		95.2	16	
	mutual_info_classif–Mutual information	95.76	14	14 (Accuracy %: 95.76)
		95.43	16	
		95.4	15	
	chi2–Chi-squared stats	96.17	34	28 (Accuracy %:96.01)
		96.09	31	
		96.02	29	
Random Forest (RF)	f_classif–ANOVA F	97.79	46	16 (Accuracy %:97.0)
		97.75	43	
		97.65	39	
	mutual_info_classif–Mutual information	97.77	46	13 (Accuracy %:97.3)
		97.76	42	
		97.73	44	
	chi2–Chi-squared stats	97.79	47	25 (Accuracy %:97.0)
		97.75	42	
		97.74	43	
Gradient Boosting (GB)	f_classif–ANOVA F	98.24	45	43 (Accuracy %:98.07)
		98.15	44	
		98.13	46	
	mutual_info_classif–Mutual information	98.27	47	36 (Accuracy %: 98.07)
		98.21	43	
		98.19	40	
	chi2–Chi-squared stats	98.26	48	42 (Accuracy %: 98.1)
		98.22	47	
		98.14	44	
Extreme Gradient Boosting (XGB)	f_classif–ANOVA F	98.1	47	43 (Accuracy %: 98.05)
		98.05	43	
		98.02	44	
	mutual_info_classif–Mutual information	98.19	44	34 (Accuracy %:98.08)
		98.15	46	
		98.14	43	
	chi2–Chi-squared stats	98.21	45	34 (Accuracy %:98.09)
		98.2	46	
		98.19	42	

Tables 14 and 15 illustrate the minimum and a maximum number of features necessary to achieve optimal accuracy for Datasets-1 and -2 using eight classification algorithms. The minimum number of features required to provide a larger accuracy margin suggests that

a dimension reduction will improve performance. Tables 16 and 17 show a summary of the overall performance in terms of accuracy for Dataset-1 and Dataset-2 with different tuning factors. Dataset-2 is already a balanced set; hence, no SMOTE procedure was necessary. Figures 8 and 9 demonstrate conclusively that tuned classification algorithms are more accurate than untuned classification algorithms. Random Forest (RF) and Gradient Boosting (GB) achieve 97.440% and 97.47% accuracy (%) for Dataset-1, respectively. For Dataset-2, Extreme Gradient Boosting (XGB) and Gradient Boosting (GB) have a higher accuracy (%) of 98.21 and 98.27, respectively.

Table 16. Dataset-1 performance (accuracy %) comparison.

Classification Algorithm	Untuned	Balanced Dataset	Hyper-parameter Tuned	Feature Selection
Logistic Regression (LogR)	92.537	92.261	92.374	92.33
Support Vector Machine (SVM)	94.744	94.949	96.898	97.04
Bernoulli Naïve Bayes (BNB)	90.285	90.596	90.531	91.27
K-Neighbors Classifier (KNN)	94.708	94.924	96.832	97.04
Decision Tree (DT)	96.182	96.694	96.475	96.690
Random Forest (RF)	97.223	97.425	97.466	97.440
Gradient Boosting (GB)	94.699	94.713	97.076	97.47
Extreme Gradient Boosting (XGB)	97.286	97.028	97.182	97.260

Table 17. Dataset-2 performance (accuracy %) comparison.

Classification Algorithm	Untuned	Hyper-Parameter Tuned	Feature Selection
Logistic Regression (LogR)	93.01	93.82	93.88
Support Vector Machine (SVM)	94.48	96.66	96.79
Gaussian Naïve Bayes (GNB)	83.73	84.4	86.99
K-Neighbors Classifier (KNN)	93.63	94.73	95.41
Decision Tree (DT)	95.57	94.52	96.17
Random Forest (RF)	97.62	97.74	97.79
Gradient Boosting (GB)	97.25	97.98	98.27
Extreme Gradient Boosting (XGB)	98.15	98.26	98.21

The performance comparison between existing approaches and our fine-tuned method is shown in Table 18 and Figures 10 and 11. The results show that the fine-tuned method outperforms with the UCI dataset and receives almost the same results as the ensemble methods with the Mendeley dataset.

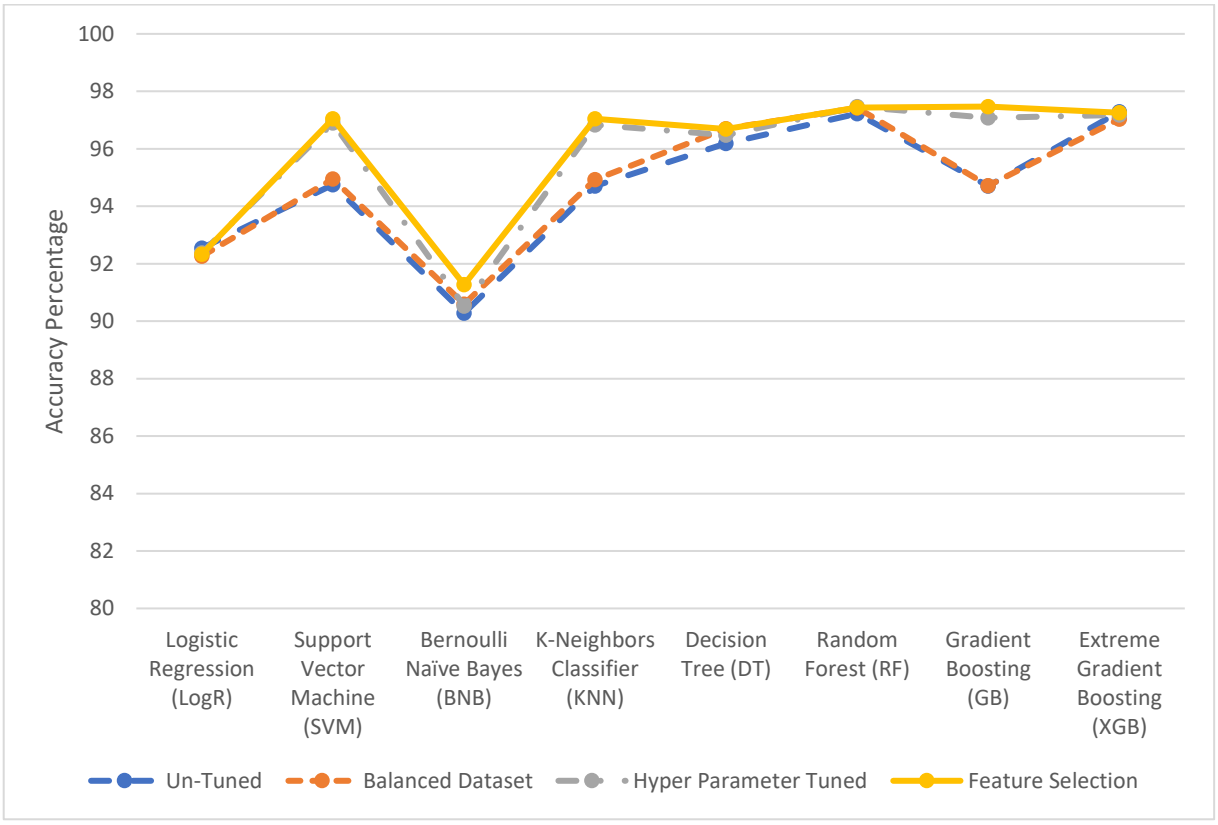

Figure 8. Dataset-1 performance (accuracy %) comparison.

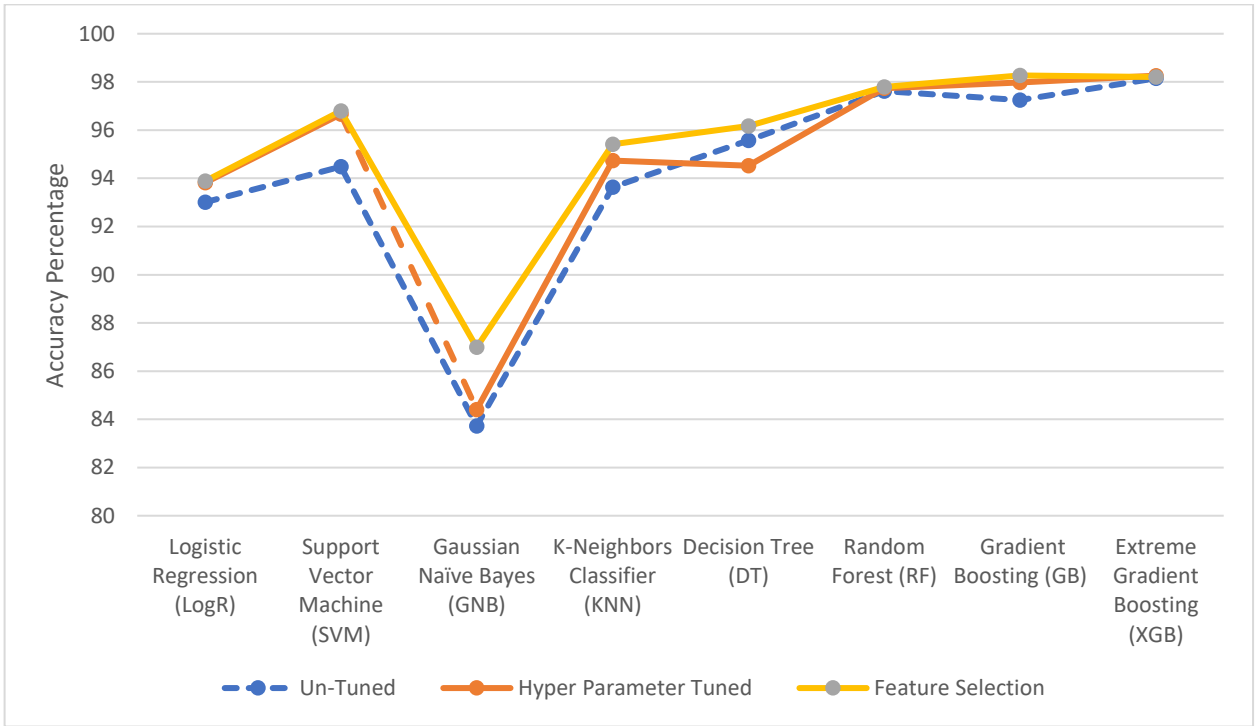

Figure 9. Dataset-2 performance (accuracy %) comparison.

Table 18. Performance comparison of existing and fine-tuned method for Dataset-1 and Dataset-2.

Author	Dataset	Algorithm	Accuracy %
KHAN., 2020 [12]	UCI	RF	97
		ANN	97
SALIHOVIC., 2019 [13]	UCI	RF with Ranker + Principal Component Optimization	97.33
		RF with BestFirst + CfsSubsEval Optimization	94.24
VISHVA., 2021 [14]	UCI	RF	97
		LogR	92
HUTCHINSON., 2018 [15]	UCI	RF- Feature Set D	95.5
		RF- Feature Set E	96.5
SARASJATI., 2022 [16]	UCI	RF	88.92
	Mendeley	RF	97.50
AL-SAREM., 2021 [19]	UCI	Optimal stacking ensemble method	97.16
	Mendeley-Dataset 1	Optimal stacking ensemble method	98.58
	Mendeley-Dataset 2	Optimal stacking ensemble method	97.39
FINE-TUNED METHOD (PROPOSED)	UCI	RF	97.44
		GB	97.47
	Mendeley-Dataset	GB	98.27
		XGB	98.21

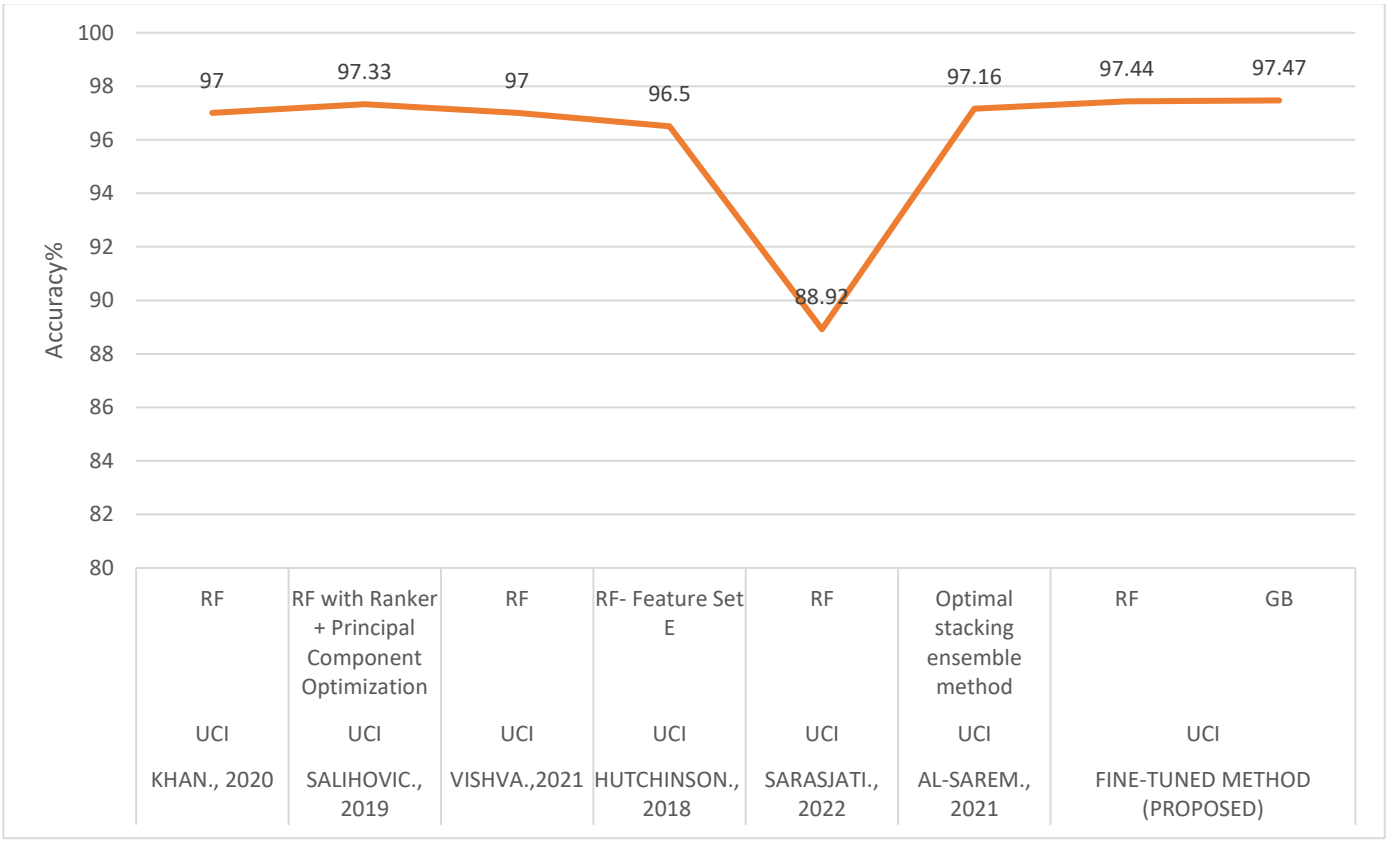

Figure 10. Performance comparison of existing and fine-tuned system for UCI dataset [12–16,19].

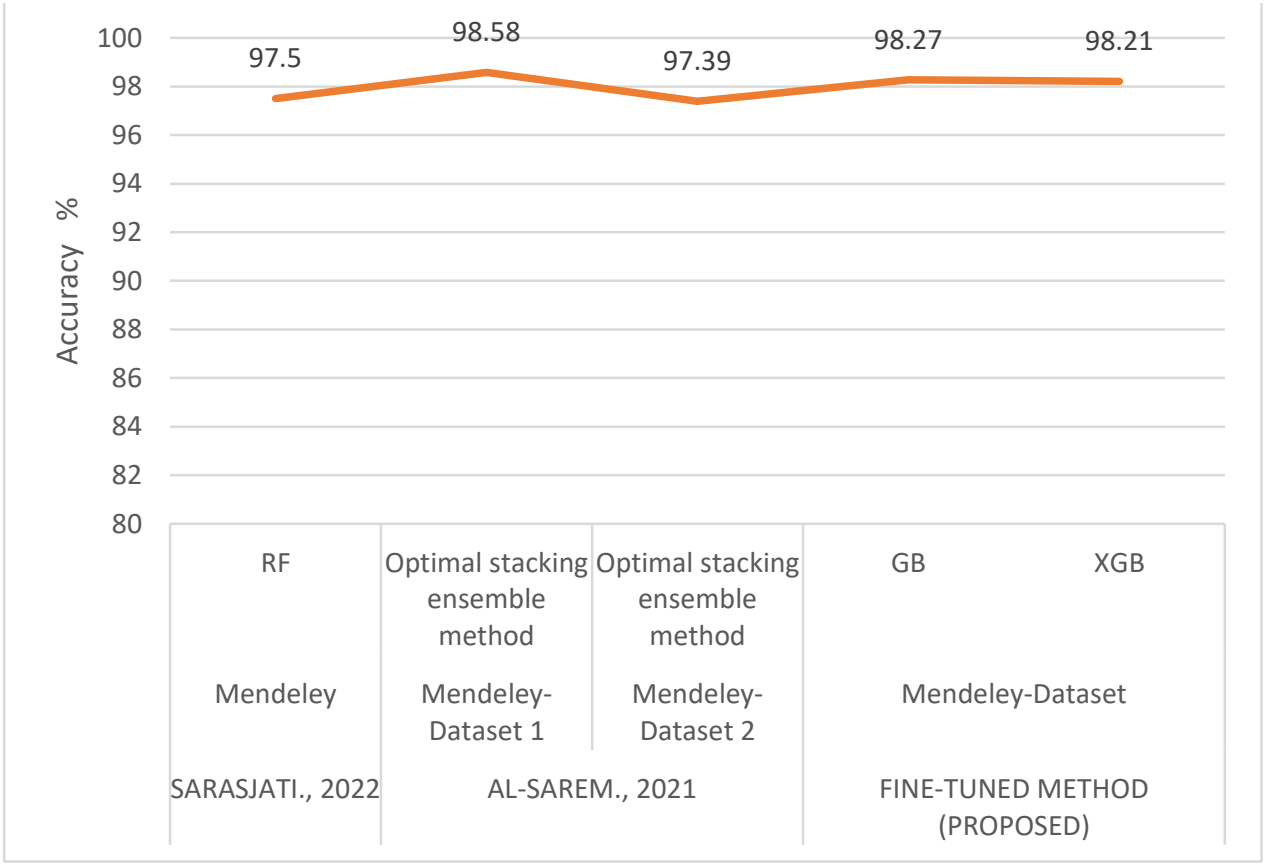

Figure 11. Performance comparison of existing and fine-tuned method for Mendeley dataset [16,19].

6. Conclusions and Future Scope

Phishing via spoofed URLs is a deceptive technique used to deceive users into downloading malicious software or disclosing personal information. Most of these phishing URLs are spread through email or SMS or through social media post. To detect these URLs is a real challenging task for internet users. The machine learning based solution provides an automatic detection of such harmful URLs. Existing research works focus on enhancing detection accuracy by the introduction of novel methods, such as selecting some features, ranking some features, grouping some features, ensemble of the machine learning model and so on. Tuning parts of the machine learning model are given less consideration.

Adding single tuning factors to the machine learning model does not create significant impact on the performance. That is why this experiment evaluates the performance of the eight machine learning algorithms with three different performance tuning factors, such as dataset balancing, hyper-parameter tuning and feature selection.

The results demonstrate conclusively that tuned classification algorithms are more accurate than existing classification algorithms. According to our investigations, data balancing results in a minor improvement in performance. In hyper-parameter tuning, the experimental results show that SVM, KNN and GB algorithms improve the accuracy when applied to Dataset-1. For Dataset-2, SVM, KNN, GNB and DT add considerable improvements. With regards to feature selection, the result highlights the minimum number of features required to achieve higher accuracy margin. This will help the future research to choose best scoring function with an optimum number of features. By combining all the three fine-tuning factors with the machine learning model without adding any additional procedure, the model performs better in terms of accuracy. The result shows that tuning factors enhance the efficiency of machine learning algorithms. For Dataset-1, Random Forest (RF) and Gradient Boosting (XGB) achieve accuracy rates of 97.44% and 97.47%,

respectively. Gradient Boosting (XGB) and Extreme Gradient Boosting (XGB) achieve accuracy values of 98.27% and 98.21%, respectively, for Dataset-2.

In the future we will work on important techniques in the network known as software-defined networking (SDN) and blockchain technology that offer several benefits by separating the intelligence of the network (the controller) from the underlying network architecture. SDN is an example of an advantage that may be gained from this separation (data plane) [35]. The most significant innovation behind Bitcoin is the blockchain innovation mechanism. It is still dealing with issues such as security, information management, compliance and dependability. In the future, our research will uncover various cloud security concerns and network issues in order to thwart cloud threats and facilitate risk reduction strategies for researchers, end-users and cloud providers undertaking threat analysis [36].

Author Contributions: Data Curation, S.R.A.S., S.B., A.S.A.-K. and B.S.; Formal Analysis, S.R.A.S., S.B., A.S.A.-K., B.S. and S.C.; Funding Acquisition, A.M., J.L.W. and A.B.; Investigation, S.C. and B.S.; Methodology, S.R.A.S., S.B., A.S.A.-K. and B.S.; Project Administration, A.M., J.L.W. and A.B.; Resources, B.S. and S.C.; Software, S.R.A.S., S.B., A.S.A.-K. and B.S.; Validation, B.S.; Visualization, A.M., J.L.W. and A.B.; Writing—Original Draft, S.R.A.S., S.B., A.S.A.-K. and B.S.; Writing—Review and Editing, A.M., J.L.W. and A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data can be publicly accessed from: https://archive.ics.uci.edu/ml/datasets/website+phishing (accessed on 26 March 2015); https://archive.ics.uci.edu/mL/datasets/Phishing+Websites (accessed on 26 March 2015).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Andress, J. *The Basics of Information Security*, 2nd ed.; Chapter 8; Syngress: Oxford, UK, 2014. [CrossRef]
2. Anti-Phishing Working Group (APWG) Legacy Reports. Available online: https://docs.apwg.org/reports/apwg_trends_report_q2_2022.pdf (accessed on 1 December 2022).
3. Raja, A.S.; Madhubala, R.; Rajesh, N.; Shaheetha, L.; Arulkumar, N. Survey on Malicious URL Detection Techniques. In Proceedings of the 6th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 28–30 April 2022; pp. 778–781. [CrossRef]
4. Raja, A.S.; Pradeepa, G.; Arulkumar, N. Mudhr: Malicious URL detection using heuristic rules based approach. In *AIP Conference Proceedings*; AIP Publishing LLC: Woodbury, NY, USA, 2022; Volume 2393.
5. Mohammad, R.; Thabtah, F.; McCluskey, T.L. Phishing Website Features. Available online: https://eprints.hud.ac.uk/id/eprint/24330/6/MohammadPhishing14July2015.pdf (accessed on 1 December 2022).
6. Raja, A.S.; Vinodini, R.; Kavitha, A. Lexical features based malicious URL detection using machine learning techniques. *Mater. Today Proc.* **2021**, *47*, 163–166. [CrossRef]
7. Hou, Y.-T.; Chang, Y.; Chen, T.; Laih, C.-S.; Chen, C.-M. Malicious web content detection by machine learning. *Expert Syst. Appl.* **2010**, *37*, 55–60. [CrossRef]
8. Raja, A.S.; Sundarvadivazhagan, B.; Vijayarangan, R.; Veeramani, S. Malicious Webpage Classification Based on Web Content Features Using Machine Learning and Deep Learning. In Proceedings of the International Conference on Green Energy, Computing and Sustainable Technology (GECOST) 2022, Virtual, 26–28 October 2022. [CrossRef]
9. Sahoo, D.; Liu, C.; Hoi, S.C. Malicious URL Detection using Machine Learning: A Survey. *arXiv* arXiv:1701.07179, 2017.
10. Awasthi, A.; Goel, N. Phishing website prediction using base and ensemble classifier techniques with cross-validation. *Cybersecurity* **2022**, *5*, 22. [CrossRef] [PubMed]
11. Tang, L.; Mahmoud, Q.H. A Survey of Machine Learning-Based Solutions for Phishing Website Detection. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 672–694. [CrossRef]
12. Khan, S.A.; Khan, W.; Hussain, A. Phishing Attacks and Websites Classification Using Machine Learning and Multiple Datasets (A Comparative Analysis). In *Intelligent Computing Methodologies: 16th International Conference, ICIC 2020, Bari, Italy, 2–5 October 2020, Proceedings, Part III*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 12465. [CrossRef]

13. Salihovic, I.; Serdarevic, H.; Kevric, J. The Role of Feature Selection in Machine Learning for Detection of Spam and Phishing Attacks. Advanced Technologies, Systems, and Applications. In *Advanced Technologies, Systems, and Applications II: Proceedings of the International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies (IAT)*; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2019; Volume 60, p. 60. [CrossRef]
14. Vishva, E.S.; Aju, D. Phisher Fighter: Website Phishing Detection System Based on URL and Term Frequency-Inverse Document Frequency Values. *J. Cyber Secur. Mobil.* **2021**, *11*, 83–104. [CrossRef]
15. Hutchinson, S.; Zhang, Z.; Liu, Q. Detecting Phishing Websites with Random Forest. Machine Learning and Intelligent Communications: Third International Conference, MLICOM 2018, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Hangzhou, China, 6–8 July 2018; Meng, L., Zhang, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 251. [CrossRef]
16. Sarasjati, W.; Rustad, S.; Santoso, H.A.; Syukur, A.; Rafrastara, F.A. Comparative Study of Classification Algorithms for Website Phishing Detection on Multiple Datasets. In *International Seminar on Application for Technology of Information and Communication (iSemantic)*; IEEE: New York, NY, USA, 2022; pp. 448–452. [CrossRef]
17. Tama, B.A.; Rhee, K.-H. A Comparative Study of Phishing Websites Classification Based on Classifier Ensembles. *J. Korea Multimed. Soc.* **2018**, *21*, 617–625. [CrossRef]
18. Karabatak, M.; Mustafa, T. Performance comparison of classifiers on reduced phishing website dataset. In Proceedings of the 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018; pp. 1–5. [CrossRef]
19. Al-Sarem, M.; Saeed, F.; Al-Mekhlafi, Z.G.; Mohammed, B.A.; Al-Hadhrami, T.; Alshammari, M.T.; Alreshidi, A.; Alshammari, T.S. An Optimized Stacking Ensemble Model for Phishing Websites Detection. *Electronics* **2021**, *10*, 1285. [CrossRef]
20. Feroz, M.N.; Mengel, S. Examination of data, rule generation and detection of phishing URLs using online logistic regression. In *IEEE International Conference on Big Data (Big Data)*; IEEE: New York, NY, USA, 2014; pp. 241–250.
21. Anupam, S.; Kar, A.K. Phishing website detection using support vector machines and nature-inspired optimization algorithms. *Telecommun. Syst.* **2021**, *76*, 17–32. [CrossRef]
22. Machado, L.; Gadge, J. Phishing Sites Detection Based on C4.5 Decision Tree Algorithm. In Proceedings of the International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 17–18 August 2017; pp. 1–5.
23. Altyeb, A. Phishing Websites Classification using Hybrid SVM and KNN Approach. *Int. J. Adv. Comput. Sci. Appl.* **2017**, 8. [CrossRef]
24. Subasi, A.; Molah, E.; Almkallawi, F.; Chaudhery, T.J. Intelligent phishing website detection using random forest classifier. In Proceedings of the International Conference on Electrical and Computing Technologies and Applications (ICECTA), Phuket, Thailand, 12–13 October 2017; pp. 1–5.
25. Bhoj, N.; Bawari, R.; Tripathi, A.; Sahai, N. Naive and Neighbour Approach for Phishing Detection. In Proceedings of the IEEE International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 18–19 June 2021; pp. 171–175.
26. Brownlee, J. *Ensemble Learning Algorithms With Python: Make Better Predictions with Bagging, Boosting, and Stacking*; Machine Learning Mastery: San Francisco, CA, USA, 2021.
27. Tougui, I.; Jilbab, A.; El Mhamdi, J. Impact of the Choice of Cross-Validation Techniques on the Results of Machine Learning-Based Diagnostic Applications. *Healthc. Inform. Res.* **2021**, *27*, 189–199. [CrossRef] [PubMed]
28. Mohammad, R.; McCluskey, T.L.; Thabtah, F. UCI Machine Learning Repository: Phishing Websites Data Set. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 26 March 2015).
29. Tan, C.L. Phishing Dataset for Machine Learning: Feature Evaluation. *Mendeley Data* **2018**, *1*, 2018.
30. Almseidin, M.; Zuraiq, A.A.; Al-Kasassbeh, M.; Alnidami, N. Phishing Detection Based on Machine Learning and Feature Selection Methods. *Int. J. Interact. Mob. Technol.* **2019**, *13*, 171–183. [CrossRef]
31. Ul Hassan, I.; Ali, R.H.; Ul Abideen, Z.; Khan, T.A.; Kouatly, R. Significance of Machine Learning for Detection of Malicious Websites on an Unbalanced Dataset. *Digital* **2022**, *2*, 501–519. [CrossRef]
32. Zheng, M.; Wang, F.; Hu, X.; Miao, Y.; Cao, H.; Tang, M. A Method for Analyzing the Performance Impact of Imbalanced Binary Data on Machine Learning Models. *Axioms* **2022**, *11*, 607. [CrossRef]
33. Synthetic Minority Over-Sampling TEchnique (SMOTE). Available online: https://medium.com/@corymaklin/synthetic-minority-over-sampling-technique-smote-7d419696b88c (accessed on 1 December 2022).
34. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
35. Badotra, S.; Panda, S.N. SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking. *Clust. Comput.* **2021**, *24*, 501–513. [CrossRef]
36. Rani, M.; Guleria, K.; Panda, S.N. Blockchain Technology Novel Prospective for Cloud Security. In Proceedings of the 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), Noida, India, 13–14 October 2022; IEEE: New York, NY, USA; pp. 1–6.