Journal of Computer Virology and Hacking Techniques (2025) 21:34
https://doi.org/10.1007/s11416-025-00579-9

RESEARCH

Securing healthcare systems: a random forest approach to malicious URL detection

Anuja Nair[1] · Pal Patel[1] · Himanshu Vadher[1] · Man Patel[1] · Tarjni Vyas[1] · Chintan Bhatt[2] · Luana Conte[3] · Giorgio De Nunzio[4]

Received: 11 March 2025 / Accepted: 16 September 2025

Abstract

Purpose: With technological advancements, uniform resource locators (URLs) are increasingly used in healthcare to store patient records, reducing paperwork. However, security concerns arise as malicious URLs can deceive users, leading to data breaches. Machine learning (ML) offers a solution by analyzing past data to predict whether a URL is malicious or benign.

Methods: In this work, a dataset from GitHub containing 151,828 URL samples was pre-processed, revealing unique characteristics of malicious URLs. Ad hoc feature extraction techniques were applied to capture these distinguishing traits. To classify URLs, various supervised ML classifiers were used, including logistic regression (LR), perceptron, decision tree (DT), random forest (RF), extreme gradient boosting (XGBoost), adaptive boosting (AdaBoost), gradient boost (GB), k-nearest neighbors (KNN), support vector machine (SVM), cat boost (CB), multinomial naive bayes (MNB), bernoulli baive bayes (BNB), light gradient boosting (LGBM) and passive aggressive classifier (PAC). Additionally, "automatic" feature extraction was performed using the term frequency-inverted document frequency (TF-IDF) method and the extracted features were then used with models such as LR, DT, RF, XGBoost, CB, KNN, LGBM, PAC, MNB, and BNB.

Results: Experimental results demonstrate that automatic feature extraction improves classification accuracy, making it a reliable method for detecting malicious URLs. The RF classifier had the best performance with both methods, achieving 99.82% accuracy with automatic feature extraction compared to 99.57% with hand-crafted features. The other metrics also improved with automatic feature extraction, including 99.84% precision, 99.44% recall, and 99.64% F1 score.

Conclusion: This approach has potential applications in securing healthcare systems, web browsers, and cybersecurity platforms, helping prevent unauthorized access to sensitive information.

Keywords Malicious URLs · Feature extraction · Healthcare · TF-IDF · Health URLs · Machine learning

✉ Giorgio De Nunzio
giorgio.denunzio@unisalento.it

Anuja Nair
anuja.nair@nirmauni.ac.in

Pal Patel
20BCE185@nirmauni.ac.in

Himanshu Vadher
20BCE092@nirmauni.ac.in

Man Patel
21BCE206@nirmauni.ac.in

Tarjni Vyas
tarjni.vyas@nirmauni.ac.in

Chintan Bhatt
cbhatt@uow.edu.au

Luana Conte
luana.conte@unipa.it

[1] Department of Computer Science and Engineering, School of Technology, Nirma University, 382481 Ahmedabad, India

[2] University of Wollongong, GIFT City Campus, Gandhinagar, Gujarat, India

[3] Department of Physics and Chemistry "Emilio Segré", University of Palermo, Palermo, Italy

[4] Department of Mathematics and Physics "Ennio De Giorgi", University of Salento, Lecce, Italy

Published online: 17 October 2025

1 Introduction

In today's user-centred technology generation, the use of web technologies in the healthcare industry has improved the quality and value of healthcare. As medical practice transitions to paperless procedures, the current generation of electronic health records (EHR) focuses on information acquisition and workflow [1, 2]. EHR helps all medical practitioners to use the same tools and interface. The concept of health uniform resource locator (URL) comes into the picture with this advancing technology. The health URL maintains the patient's health record and makes it easier for the doctors to understand the patient's health condition through this record [3, 4]. Every patient is assigned a URL in which their health records are kept, and the URL is protected with a password whose access is given only to authorized parties. A significant problem with this health URL is its privacy and security issues [5–7]. An attacker can manipulate the health URL to create a fake authentication page to steal the password and patient's health information. Such security challenges in critical systems like healthcare have been extensively studied by recent researchers [8] and it is evident that special procedures need to be employed in order to protect sensitive information. The malicious health URL can become a gateway to hack the user's system and steal the data. To prevent this threat, it becomes essential for us to identify malicious URLs and protect our system.

Detection of these threats and providing security against them is a serious issue [9]. The blacklist approach is the conventional technique for identifying malicious URLs [10]. In this approach, a large list of malicious URLs is maintained. It is an effective solution for finding malicious URLs with slow growth in malicious URLs. However, with rapid growth in malicious URLs, updating the blacklist frequently becomes difficult, and it fails to detect malicious URLs effectively [11, 12]. Blacklist methods are static, do not have learning capability, and can be easily breached by the attackers through continuous testing [13, 14]. Machine learning (ML) makes this job easy as it provides various models through which accurate predictions can be made by learning from past experiences [15]. The drawbacks of the blacklist method give more importance to ML techniques [16]. Recent research into optimization algorithms, including nature-inspired approaches [17], has explored various classification techniques in cybersecurity that highlight the necessity for adaptive and dynamic solutions. Our ML-based approach addresses these same challenges through robust feature engineering and classifier selection. Large amounts of data may be used to train ML systems to forecast new URLs. Because of these characteristics of ML algorithms, it is widely applied by researchers in the field of cyber security, including anomaly detection, harmful URL categorization [18], intrusion detection, etc [19].

The ML techniques used here include data collection, feature engineering, data pre-processing, model creation, testing, and classification. Feature engineering includes the process of feature extraction and selection [20, 21]. Feature engineering transforms the data into unique features describing the data effectively [22]. While there are a number of approaches for feature engineering in security contexts, including those that incorporated chaotic optimization techniques [23], our method is focused on finding the most valuable URL features for maximizing detection rates with minimal computational cost. It helps optimize the model performance on unknown data [8, 24]. The amount of data, processing and sample quality affect how well a detection is done [25]. Our work addresses malicious URL detection as a binary classification problem and studied the performance of logistic regression (LR), perceptron, decision tree (DT), random forest (RF), extreme gradient boosting (XGBoost), adaptive boosting (AdaBoost), gradient boost (GB), k-nearest neighbors (KNN), support vector machine (SVM), cat boost (CB), multinomial naive bayes (MNB), bernoulli baive bayes (BNB), light gradient boosting (LGBM) and passive aggressive classifier (PAC), the classification ML algorithms on the dataset [26]. RF in particular has shown strong predictive capabilities in diverse applications ranging from environmental resource management [27] to security contexts like ours. Using this data and the extracted features, we trained the ML classifiers to learn specific patterns, similarities or attributes that distinguish malicious and non-malicious URLs.

1.1 Motivation

In the current world where technology plays an important role in every field, it has its own set of duties in the healthcare industry. The healthcare industry is seeing constant digitization in order to relinquish the cumbersome paper work and file maintenance and switching this the electronic media known as EHRs. These EHRs are digital records of a person's medical information, their treatment history and their past diagnosis details. They carry a huge amount of sensitive information in them and when in the wrong or irresponsible hands, it can create havoc for the patient to whom the EHR belongs. The EHRs are passed on to doctors and patients through the use of URLs. But it might be the case that the URL is fake and that it tries to get the login details of a patient's EHR in order to use the information in a wrongful manner. It might be the case that after the EHR is accessed through the misleading URL, the data is altered in such a manner that it proves to be dangerous for the patient. Consider a realistic attack scenario where a patient receives

an email that appears to be came from its healthcare provider with a URL to view their recent lab test results. The email appears to be legitimate because of the hospital's branding and mention of their recent visit to the hospital. when the patient clicks on that malicious URL, it leads to a fake website that resembles the hospital's EHR portal interface. When the patient enters their credentials, the attacker not only captures their log-in information, but also gains the ability to modify their medical records. In a more severe case, an attacker could access hospital's internal networks and insert harmful URL into genuine patient communication channels, such as appointment reminders or prescription notifications. Such attacks could lead to unauthorized access to sensitive medical information, identity theft, insurance fraud, or even life-threatening situations where critical medical data is altered or deleted. Such scenarios highlight the urgent need for automated detection mechanisms that can identify malicious URLs before they reach healthcare stakeholders. All these things are the reason why there is a need to be aware about the URL that is being tapped upon by the user is malicious or safe to go forward with. Therefore, this model has been trained to carry out the task of classification of URL into malicious or non-malicious.

1.2 Research contributions

The main contribution of this article can be summarised as follows:

- To propose a novel approach for malicious URL detection using RF classifier for securing EHR data from attackers that can manipulate health URL.
- To handle the problem of interference of human variables using feature engineering classifiers such as frequency-inverted document frequency (TF-IDF).
- To compare the handcrafted and automatic feature extraction process.
- To evaluate the proposed architecture using different evaluation metrics, such as accuracy, precision, recall, and F1 score using various ML models.

1.3 Paper organization

The rest of the article has its section outline: Sect. 2 provides a literature review on the documents related to malicious URLs. Sect. 3 presents the problem formulation, defining the mathematical framework for URL classification. Sect. 4 gives an overview of the methodology followed. In Sect. 5, the experimental outcomes have been discussed. Section 6 offers the conclusion of our work.

2 Related work

Malicious URLs pose a significant security threat as they are commonly used to launch cyber attacks and steal user information. Recent studies on malicious URL detection are described in this section. The work by Nowroozi et al. [28] develops ML models to identify fraudulent advertisement URLs using lexical and web-scraped features. While they achieve decent detection accuracy of 99.63% with XGBoost, the feature engineering relies predominantly on URL strings lacking content analysis. Furthermore, the ensemble classifiers demonstrate high vulnerability against ZOO exploratory attacks with up to 96.67% success rate against RF. Practical implementation challenges, including performance retention over time, still need to be addressed (Table 1).

In [11], the authors have presented a technique to detect the malicious URL more accurately by dividing it into three classes: phishing, legitimate homepage, and legitimate login. They used yearly data sets illustrating how the ML models reduce their accuracy as a function of time by training the models with previous data sets, testing them with new URLs, and comparing ML and deep learning methodologies. Its applications detect legitimate login pages, legitimate homepages, and phishing sites. They obtained an accuracy of 96.50% using TF-IDF in conjunction with N-gram and LR. In [12], the authors have detected malicious URLs robustly and effectively, reducing the uncertainty of URL tokens and learning the long-distance reliance on URL tokens. Their testing results showed that the self-paced comprehensive and deep solution outperforms other solutions in terms of robustness and efficiency in detecting dangerous URLs.

In [33], they had worked on solving the category imbalance problem and detecting malicious URLs by using the CS-XGB method. In comparison to SMOTE + XGB, which uses SMOTE to cope with variable data before constructing the XGB model, CS-XGB functions well and saves both time and space. However, future modelling will be complex using this method as the data increases. In [13], the authors have proposed a method aiming to improve classification accuracy, as well as the ability to analyze and detect malicious URLs. Using capsule networks (CapsNet) and independently recurrent NNs (IndRNN), they created a parallel joint neural network model capable of capturing the semantic and visual information of URLs. This model focuses on critical traits that improve classification accuracy. According to their findings, CapsNet outperforms convolutional NNs (CNNs) in extracting features from images, and IndRNN outperforms long short-term memory (LSTM) networks in robustness for sequence learning tasks. Their method enables the simultaneous acquisition of a URL's semantic

Table 1 A comparison of state-of-the-art approaches for malicious URL detection. Acc = accuracy, Prec = precision, Rec = recall, F1 = F1 score

Author	Year	Objective	Technology Used	Pros	Cons	Acc(%)	Prec (%)	Rec (%)	F1 (%)
Proposed Architecture	2025	To enhance the application of ML in cyber security by raising the accuracy of ML models used for detecting malicious URLs	RF	Avoids interference of human variables by using feature engineering classifiers	-	99.82	99.84	99.44	99.64
Maftoun et al. [29]	2024	To detect malicious websites using ML techniques, with a focus on optimizing the Hist GB classifier (HGBC) using grid search	HGBC, synthetic minority over-sampling technique (SMOTE)	Addressed class imbalance using SMOTE	Computational intensity of grid search	96	96	96	96
Reyes-Dorta et al. [30]	2024	To analyze the application of different ML techniques, including quantum ML, for the early detection of fraudulent URLs	LR, DTs, SVMs, neural networks (NNs), variational quantum classifier (VQC)	Compared performance of classical and quantum ML approaches	Challenges in encoding alpha-numeric variables for quantum ML processing	97	–	–	–
Nowroozi et al. [28]	2023	Analyse adversarial attacks on ML models for malicious URL detection	RF, GB, XGBoost, AdaBoost, k-means	Achieves high detection accuracy	Models vulnerable to exploratory attacks	99.63	-	-	-
Abad et al. [31]	2023	Classify malicious URLs using ML models	SVM, RF, DTs, KNN	Instance selection methods improve efficiency	SVM has high training time; Phishing URLs are challenging to classify	-	93.19	91.19	92.18
Rafsanjani et al. [32]	2023	Develop a secure QR code scanner for detecting malicious URLs	Blacklist, lexical, host-based and content-based feature classification	Privacy-friendly app with least privileges	Scope for improving accuracy by prioritizing features	93.5	93.8	–	–
Sanchez-Paniagu et al. [11]	2022	To detect the malicious URL more accurately by dividing it into three classes: phishing, the legitimate homepage, and legitimate login and to show how models decrease their accuracy over time	LGBM, XGBoost, KNN, AdaBoost, RF, SVM, NB, LR	The model trains on actual login pages, not just homepage URLs	Overall accuracy is reduced due to the similarity between phishing and legitimate samples	96.50	96.54	96.48	96.51
Liang et al. [12]	2022	To detect malicious URLs robustly and effectively, reduce the ambiguity of URL tokens. To learn the long-distance dependency among URL tokens	Blacklist-Based Methods, DT, SVM, KNN, NB, RF, LR, Neural networks	Self-paced learning strategies improved the robustness of the model	No analysis is provided on how robust the approach is against adversarial attacks - URLs specifically crafted to evade detection	99.27	97.42	95.44	96.38
He et al. [33]	2021	To solve the category imbalance problem and detect malicious URLs	cost-sensitive XGBoost (CS-XGB), XGBoost, SMOTE	It solves the imbalanced data problem by using CS-XGB method	Training and detection can improve, but more data may hinder future modeling	–	–	93.88	–
Chiramdasu et al. [19]	2021	To detect the existence of malicious URLs using LR	LR, KNN, SVM, LDA	The recursive detection of malicious URL increased the perfromance	Only few ML models were tested. Other ML models might provide better results	93	–	86	93

Table 1 (continued)

Author	Year	Objective	Technology Used	Pros	Cons	Acc(%)	Prec (%)	Rec (%)	F1 (%)
Indrasiri et al. [34]	2021	To propose a unique, robust ensemble ML model architecture that provides the highest prediction accuracy with a low error rate while proposing few other robust ML models	ERG-SVC model, Stacking classifier, RF, XGBoost, AdaBoost, GB, DT, KNN, LR	The model is dynamic and expandable	Limitations in feature extraction and Complex architecture	98.27	97.85	98.61	98.23
Yan et al. [24]	2020	To propose an unsupervised learning algorithm that learns URL embedding. They explored domain embedding model to obtain a good effect on domain features	SVM, DT, LR, NB, CNN	The UE classifier significantly improves malicious information detection over feature engineering	Complex and large memory needed to store high-dimensional domain embeddings	–	91	93	93
Manyumwa et al. [25]	2020	The detection of malicious URLs using URL-based features in a multiclass classification setting	XGBoost, AdaBoost, LGBM, CB	Uses only URL-based features for detection which is faster and more efficient than relying on external services or content analysis	Dataset sizes between different attack types are imbalanced which could bias models	96.98	94.47	91.32	93.83
Kumar et al. [35]	2020	To compare different ML techniques for the phishing URL classification task	LR, DT, RF, KNN, Gaussian NB	Achieves good accuracy and the methodology is language and domain-independent	More features can be experimented with leading to improving further the accuracy of the system	98.03	100	96	98

and visual components but does not support multiple classifications. In [19], the authors have developed a method to identify malicious URLs using LR and comparing other ML classifiers. For performing an experimental evaluation, they utilized more than 32,000 URLs gathered independently from Phish Tank, Kaggle, and Github to identify URLs as malicious or benign with an obtained accuracy of 91% in LR and 93% in KNN.

In [34], the authors' primary goal was to offer a new, robust ensemble ML model that provides the most remarkable prediction accuracy with the lowest error rate while suggesting a few alternative powerful ML models. They created an ensemble model that employed a voting classifier called enhanced random gradient support vector classifier (ERG-SVC) and outperformed all other models, with a prediction accuracy rate of 98.27%. The limitation of this model was its complex architecture and cloud deployment cost. In [24], the authors opted for a URL Embedding approach using unsupervised ML algorithms. They investigated specific critical characteristics of a domain embedding model to get a positive influence on domain features. In [25], a multi-class classification was performed by feeding the classification models with URL features. They included priority features like kullback–leibler divergence (KLdivergence), bag of words segmentation, and other word-based features that had shown superior performance compared to studies without those attributes. They trained those models, namely XGBoost, AdaBoost, LGBM, and CatBoost, which returned an overall accuracy above 95%. Similarly importance of feature selection has been observed in other domains, with studies like [27] demonstrating how RF models can effectively handle complex feature relationships to produce reliable predictions. In [35], a good accuracy of 98% was achieved compared to other ML models when the naïve bayes (NB) classifier was used for phishing URL classification.

Even as ML models have proven effective in detecting malicious URLs, it is a complicated process to implement these models into regular security systems, particularly in healthcare environments. System optimization studies in recent years emphasize the use of holistic approaches that bring technical solutions together with strategic deployment [8, 17, 23, 36], parallel to how security systems must achieve a balance between detection accuracy, computational cost and responsiveness to emergent threats. Our model comparison reflects this challenge by comparing various ML models to find the most suitable methods that has optimal balance of accuracy and computational cost in identifying malicious health URLs. These integrated methods are required in healthcare settings, where system dependability is of prime importance and illegal access to patient information can have severe consequences.

3 Problem formulation

This section defines the problem statement and specifies the objective function for the proposed model. Let $U = u_1, u_2, \ldots, u_n$ be the set of all URLs that users may encounter, where *n* is the total number of URLs. Each u_i has an associated binary label $y_i \in 0, 1$, where $y_i = 1$ indicates u_i is a malicious URL and $y_i = 0$ indicates u_i is a benign URL. We extract features from each URL to generate the feature set $F = \{f_1, f_2, \ldots, f_m\}$, where *m* is the total number of features and $1 \leq j \leq m$.

A function *h*() extracts features from each URL:

$$h : U \rightarrow F \tag{1}$$

Hence, for each URL u_i, we have:

$$h(u_i) = X_i = [x_{i1}, x_{i2}, \ldots, x_{im}] \tag{2}$$

where, X_i is the feature vector for URL u_i containing values for all *m* features.

The set containing feature vectors for all URLs is represented as:

$$X = \{X_1, X_2, ..., X_n\} \tag{3}$$

We aim to train a classification model $f_\theta(X_i)$ parameterized by θ that predicts the probability $\hat{y}_i$ that a given URL u_i is malicious. The classification model $f_\theta(X_i)$ takes a URL's feature vector as input and outputs a predicted probability $\hat{y}_i$ that the URL is malicious:

$$f_\theta : X_i \rightarrow \hat{y}_i \tag{4}$$

Specifically, we would like to optimize the parameters θ to minimize the cross-entropy loss over the entire dataset:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{5}$$

Additionally, we want to restrain the false positive rate, i.e. the proportion of benign URLs falsely classified as malicious, to be under some threshold *t*:

$$\frac{1}{m} \sum_{i:y_i=0} \Upsilon(\hat{y}_i > 0.5) \leq t \tag{6}$$

m is the number of benign URLs and $\Upsilon(\cdot)$ is the indicator function. The optimization problem then becomes:

$$\begin{aligned} &\min_\theta \frac{1}{n} \sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \\ &\text{s.t.} \ \frac{1}{m} \sum_{i:y_i=0} \Upsilon(\hat{y}_i > 0.5) \leq t \end{aligned} \tag{7}$$

The above equation shows that the minimization function is subject to the constraint that the false positive rate $< t$.

4 Proposed architecture

Classifying a URL as malicious or non-malicious requires a number of operations from the moment the URL is captured until the URL is classified. These operations are depicted in the proposed architecture Fig. 2, and the proposed architecture overview is represented by the system model in Fig. 1. According to the system model, the health URL is received and checked to see whether it is malicious or not using the trained ML model. Moreover, once the trained model gives the output as non-malicious, it acts as a green flag for opening the health record since it is entirely safe. The proposed architecture mainly consists of 3 stages. These stages are named the data, intelligence, and application layers. The data layer comprises four checkpoints that involve the acquisition of the URL to test for, identifying the structure of a URL, feature extraction and further data pre-processing depending on the type of features to be used in the proposed model.

Following the data layer is the intelligence layer that takes the output of the data layer as its input, which includes several feature vectors that have been extracted using various data preprocessing and feature extraction methods. The intelligence layer includes three significant steps. These steps involve getting the feature vectors, which are the output of the data layer, ready for input into the ML model, followed by the prediction by the ML model on whether the input URL is malicious or not. These three stages mark the working of the intelligence layer. Moreover, this intelligence layer is followed by the application

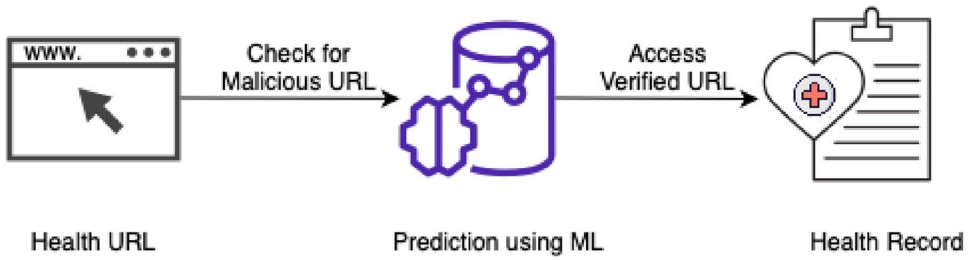

Fig. 1 System Model for Secure Health URL Access

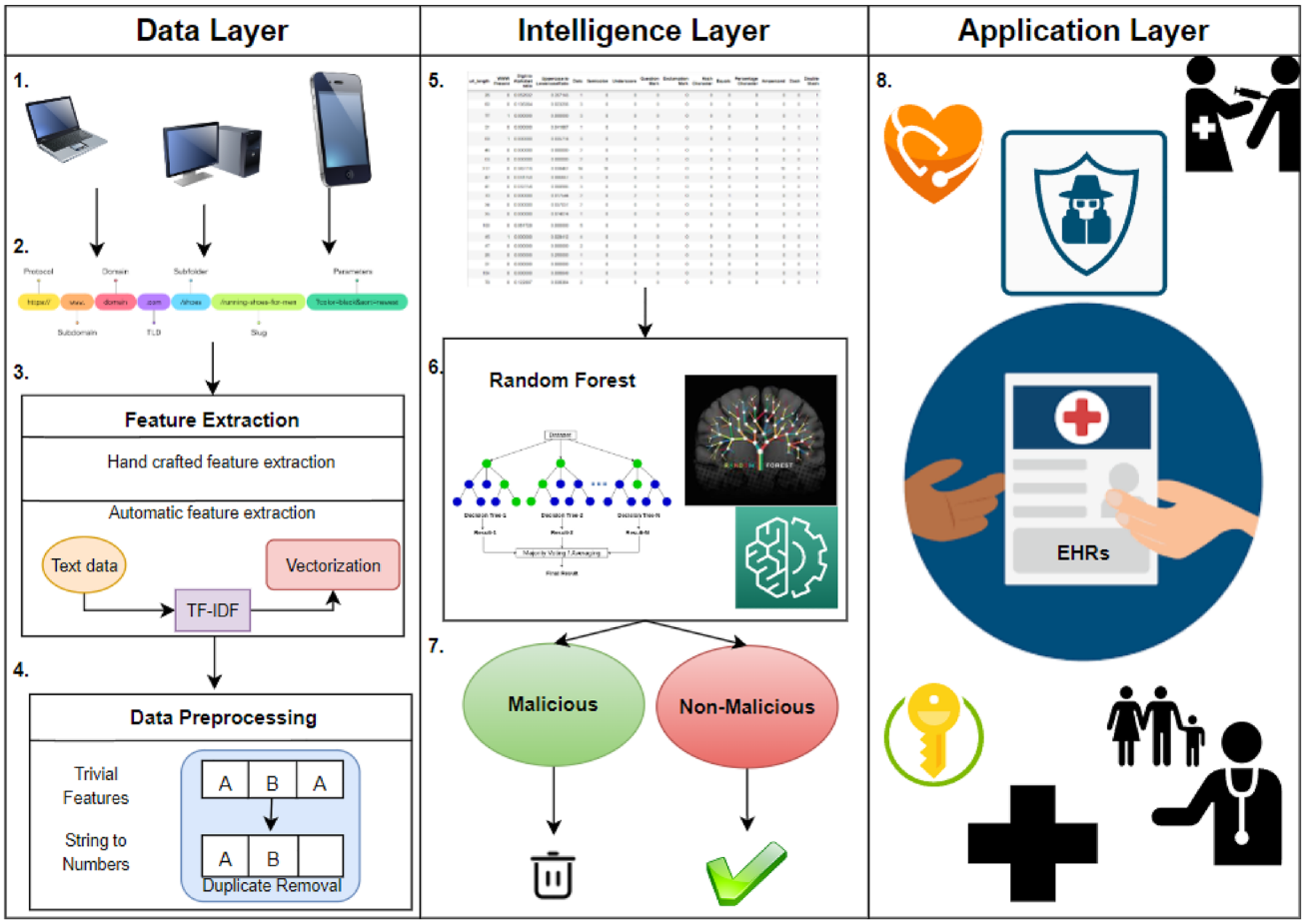

Fig. 2 Proposed Architecture for Malicious URL Detection in Healthcare Systems

layer. The output received by the intelligence layer affects the actions taken in the application layer. If the intelligence layer throws output as malicious, the health record URL is discarded; conversely, if the prediction appears non-malicious, the EHR is accessed. Algorithm 1 comprises the entire working of malicious URL detection as described in the proposed architecture.

4.1 Data layer

The initiation of the proposed architecture is done through the data layer. The data layer takes up the responsibility of the data acquisition, feature extraction and pre-processing tasks. This layer acquires the data and makes it ready for feeding into the intelligence layer with multiple methods.

4.1.1 Dataset description

The data set for training the models has been taken from [26]. The data set contains 151828 samples of URLs, of which 121462 were used for training, and 30366 were used for testing in the ratio of 4:1.

$$Dataset \xrightarrow[(80:20)]{\text{split}} train, test \tag{8}$$

Moreover, two more datasets have been used:

1. Dataset-2 [37]:- The name of the dataset is PhiUSIIL Phishing URL. The dataset contains a list of 134,850 legitimate or non-malicious URLs and 100,945 phishing or malicious URLs. The creators of the dataset have already extracted the features from the dataset and given in the file, however, those features haven't been utilized and the features enlisted below have been utilized.
2. Dataset-3 [38]:- The name of the dataset is Malicious URLs dataset present on Kaggle. The dataset contains a huge URL record of 6,51,191 URLs. Out of these given numbers 4,28,103 are benign or safe URLs, 96,457 defacement URLs, 94,111 phishing URLs, and 32,520 malware URLs. Among the given list, benign URLs have been considered as non-malicious URLs, while defacement URLs, phishing URLs and malware URLs have been included in the class of malicious URLs and therefore, labelled so.

4.1.2 Feature extraction

Feature extraction is one of the most important steps in ML. There are multiple methods for fulfilling the task of feature extraction.

- Hand Crafted Feature Extraction: The feature extraction involves extracting different features from the given URL strings from the data set. These features have been manually extracted by rule based coding. The rule based features act as the parameters on which the classification models have been trained upon. Thus, making up a parameter vector for each dataset. Moreover, these parameters have been kept as integral types for smoother model training. The features extracted are as follows:

  1. url_length: The length of the URL string. Data type-integer(int).
  2. WWW Present: Whether 'www' or 'WWW' is present in the URL. Data type-int(0-False, 1-True).
  3. Digit to Alphabet ratio: The ratio of the number of numeric characters in the URL and the number of alphabetic characters in the URL. Data type-float.
  4. UpperCase to LowerCaseRatio: The ratio of the number of upper case letters to that of the lower case letters in the URL. Data type-float.
  5. Dots: The number of dots(.) present in the URL. Data type-int.
  6. Semicolon: The number of semicolons(:) present in the URL. Data type-int.
  7. Underscore: The number of underscores(_) present in the URL. Data type-int.
  8. Question Mark: The number of question marks(?) present in the URL. Data type-int.
  9. Exclamation Mark: The number of exclamation marks(!) present in the URL. Data type-int.
  10. Hash Character: The number of hash characters(#) present in the URL. Data type-int.
  11. Equals: The number of equals to the (=) symbol in the URL. Data type-int.
  12. Percentage Character: The number of percentage signs(%) present in the URL. Data type-int.
  13. Ampersand: The number of ampersand signs(&) present in the URL. Data type-int.
  14. Dash: The number of dashes(-) present in the URL. Data type-int.
  15. Double Slash: The number of double slashes(//) present in the URL. Data type-int.
  16. Https in URL: Whether 'https' is present in the URL. Data type-int(0-False, 1-True).

- Automatic Feature Extraction: The term TF-IDF algorithm is a popular feature extraction technique. In TF-IDF, the URLs have been split into tokens or words, and the frequency of the words in the dataset is calculated. Each word is quantified and is assigned a weight according to its frequency. It yields the frequent word, a low weight and the uncommon word, a high weight. The URL is transformed into a vector, thus allowing us to feed numerical values instead of textual values into the ML models for training. Therefore, the integral vector has been created using the TF-IDF value for each given word and passed into the classifier for training. The frequency of a term in the dataset is counted as:

$$TF = \frac{\zeta}{\delta} \tag{9}$$

Where ζ is the frequency of a specific term in the dataset and δ is the total number of terms in the dataset. Inverse Document Frequency measures the importance of a term in the entire dataset. It is calculated as:

$$IDF = \log\left(\frac{\eta}{\tau}\right) \tag{10}$$

- η is the total number of URLs in the dataset and τ is the number of URLs containing the specific term. The TF and IDF values are determined to calculate the overall importance of a term in a URL relative to the entire dataset.

$$TF - IDF = TF * IDF \tag{11}$$

The TF-IDF score will be higher for the rare terms and will be lower for frequently appearing terms in the dataset.

4.1.3 Data preprocessing

The features extracted using Hand Crafted Feature Extraction method are already in numeric form. These include characteristics like URL length, presence of specific characters, and various ratios. Since they are already numeric, these features can be directly used for training ML models without additional preprocessing. The Automatic Feature Extraction method produces features in the form of tokens which further requires processing before they can be used as input for ML models. The processing involves two key steps: i) Converting string tokens to numerical form: ML models can only process numerical inputs, not strings. Therefore, the token features are transformed into numerical vectors using TF-IDF vectorization. ii) Removing duplicate instances. After conversion to numerical form, any duplicate data points are removed from the dataset. This will ensure model generalize well, prevents bias that could be introduced by repeated instances and ensures each data point contributes unique information to the models.

After eliminating duplicate columns from the data, the presence of trivial features needs to be checked using the correlation matrix. The Fig. 3 presents the heatmap of the correlation matrix. The trivial features are listed out using this because they usually correlate highly with a few other features. Therefore, these features would only add to the processing time and won't do much good in the data. After checking the correlation matrix, the p values for all the features were reviewed. Each feature has a p-value approximately equal to zero. The result is that no feature has been removed due to the p-value. Thus, no extracted features need to be dropped as an outcome of the feature selection process. Moreover, the trivial features checking has been carried out. Now, the vectors are ready to be passed as input into the proposed ML model.

4.2 Intelligence layer

The intelligence layer comprises the proposed ML model. The intelligence layer takes input as the feature vectors output from the data layer. These feature vectors are passed in as input for the proposed ML model such that it gives the output as malicious(1) or non-malicious(0) making this a binary classification problem. The proposed ML model uses the following set of equations for the processing of the input feature vectors:

To measure the impurity or randomness in the dataset, the Gini Index has been calculated as follows:

$$GiniIndex = 1 - \sum_{i=1}^{k} (P_i)^2 \tag{12}$$

Here, P_i is the probability of occurence of the class i and k is the number of classes. The entropy $E(s)$ of the dataset is calculated as:

$$E(s) = -\sum_{i=1}^{k} (P_i) \log (P_i) \tag{13}$$

$$k \rightarrow \text{ number of classes} \tag{14}$$

$$D_1, D_2, D_3, ...D_n \subset D_f \tag{15}$$

Here, D_f represents the feature vector.

$$n \rightarrow \text{ number of subests of } D_f \tag{16}$$

$$O_1 \leftarrow DT(D_1) \tag{17}$$

$$O_2 \leftarrow DT(D_2) \tag{18}$$

$$O_n \leftarrow DT(D_n) \tag{19}$$

Equations. 17, 18, and 19 represent the outputs of running a DT algorithm on subsets of the feature vectors D_f, where

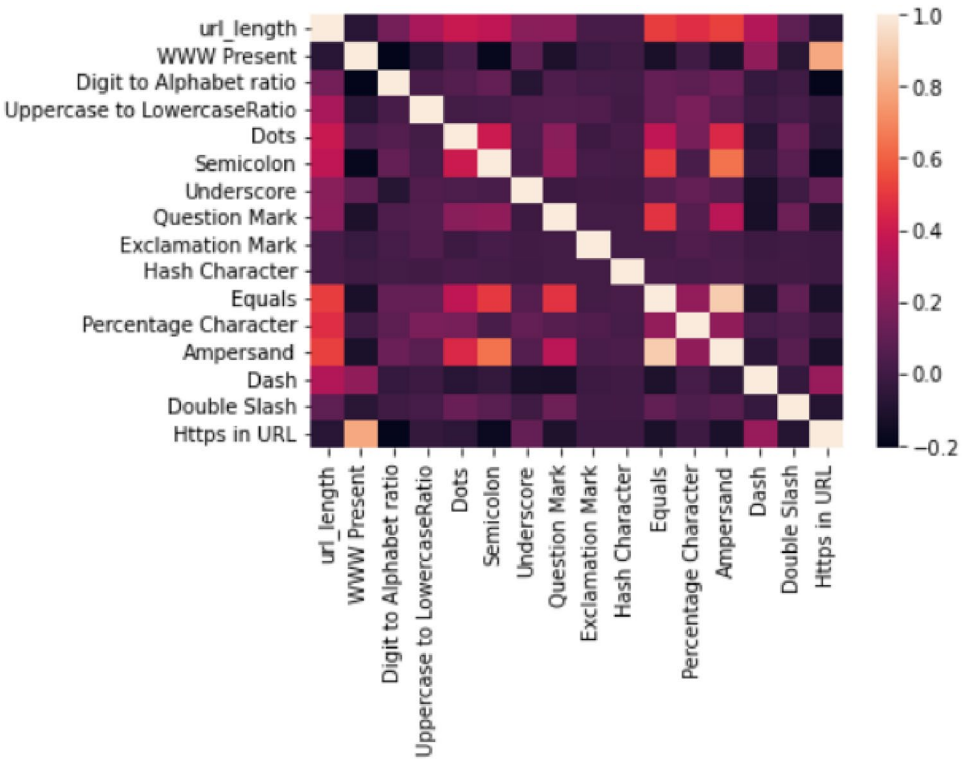

Fig. 3 Correlation Heatmap of URL-based features

each subset is denoted by D_i and DT is a function that generates a DT output.

$$Output \leftarrow Mode\,(O_1, O_2, O_3, \ldots, O_n) \tag{20}$$

This equation determines the final output class (malicious or non-malicious) by taking the mode of the outputs obtained from the DTs applied to various subsets of the feature vectors. This gives the output as one of the two classes (malicious or non-malicious). Algorithm 1 shows the entire working of URL-RF classifier interaction.

Require: $U_1, U_2, U_3, \ldots, U_n \in D_f$, $CL_0, CL_1 \in D_f$
Ensure: $Classification of CL_i$

procedure URL CLASSIFIER(D_f, RFC)
  if MissingValues $M_i \in D_f$ Dataset **then**
    $D_f Dataset \leftarrow D_f Dataset - M_i$
  else
    D_f Dataset Unaffected
  end if

  for $U_i \in D_f Dataset$ **do**
    $U_i \leftarrow U_i.split('/')$
    for $S_i \in U_i$ **do**
      $S_i \leftarrow S_i.split('-')$
      for $l_i \in S_i$ **do**
        $U_i \leftarrow U_i + l_i$
        $U_i \leftarrow U_i - S_i$
      end for
    end for
    for $S_i \in U_i$ **do**
      $S_i \leftarrow S_i.split('.')$
      for $l_i \in S_i$ **do**
        $U_i \leftarrow U_i + l_i$
        $U_i \leftarrow U_i - S_i$
      end for
    end for
    if 'com' $\in U_i$ **then**
      $U_i \leftarrow U_i$ - 'com'
    else
      U_i Unaffected
    end if
    $U_i \leftarrow$ TF-IDF Vectorizer(U_i)
    $D_f \xrightarrow{split} \zeta_{tr}, \zeta_{ts}$
    $RF \xrightarrow{applied} \zeta_{tr}$
    for $a_1, a_2, a_3, ..., a_i \in \zeta_{ts}$ **do**
      $a_i \subseteq U_s \rightarrow \zeta'_{tr}(generatesmalltrainingsamples)$
      $RF\,(\zeta'_{tr1}, \zeta'_{tr2}, ..., \zeta'_{tri}) \xrightarrow[Lowbias]{Highvariance} CL_i$
    end for
  end for
end procedure

Algorithm 1 URL Classification Using Random Forest (RF) classifier

4.3 Application layer

The application layer marks the end of the data processing part. After finishing data acquisition, feature extraction, data pre-processing, model training and class prediction, the question arises of using such a trained model. This answer is passed on by the application layer, which, in this case, talks about the employment of the classifier in the healthcare domain. Several EHRs need to be protected from outside influences, and additionally, users need to be saved from several malicious URLs passed as health records. Users may access the malicious URL and pass the login details on the malicious URL. This way, the user compromises the details to the attackers. To prevent the same, the trained model can be used to protect the patient's private details and not be misused. To demonstrate the practical implementation of this detection system, we present specific usage scenarios for different healthcare stakeholders:

1. **Patient Interface:** Patients can use this detection system either from a browser extension or a mobile application that automatically scans URLs before opening them. When a patient clicks on a URL received via email, SMS, or through patient portal notifications, the URL detection system will perform real-time analysis and will display a warning if the URL is classified as malicious. For example, when a patient clicks on a link claiming to have lab test results, their device will first verify whether it is an authentic URL or not, so attackers will not be able to steal their data, and their personal health information will remain secure.
2. **Healthcare Provider:** Physicians and Doctors can leverage this malicious URL detection system alongside their existing EHR systems and all communication channels and portals. When doctors need to send links to patients via email to book an appointment, view test results, or access prescription history, the detection system would screen those links before sending them. This system can also be installed on hospital email systems to scan for all emails containing healthcare-related links to ensure that only authenticated links are sent to patients.
3. **Healthcare System Administrator:** System administrators can install the malicious URL detection system at the network level, to secure the entire healthcare organization. The detection system can monitor all URL traffic within the hospital network, flagging suspicious links in real-time and generating alerts for security teams. This system can also be used by the administrators to analyze historical URL access patterns, identifying potential security issues and providing baseline security checks for continuous URL monitoring and compliance reporting.

This malicious URL detection system provides end-to-end protection at each touch point, wherever the malicious URLs pose a threat to healthcare data security and patient safety.

5 Results and discussion

A feature extraction process and experiments on ML classifiers have been followed. These experiments were carried out using sci-kit-learn[39] and Python. Here, we report the performance of the ML classifiers using the evaluation metrics. Results here show that processes carried out here can be a viable choice for precisely classifying URLs.

5.1 Experimental setup

The proposed model TF-IDF + RF is implemented on the Jupyter notebook in HP Pavilion 15 with Windows 10 operating system, utilizing Python version 3.7, along with the implementations of the rest of the ML models enlisted in Table 2. The machine has an Intel Core 8th generation i5-8300 H CPU at 2.30GHz, a 64-bit operating system and an x64-based processor. The PC has 8 GB of RAM, 1 TB of HDD and two graphical processing units (GPU), namely Intel(R) UHD Graphics 630 and NVIDIA GeForce GTX 1050. Multiple libraries and application programming interfaces (APIs) have been used to gain the results obtained. The data in the comma-separated value (CSV) files have been read and processed using pandas (version 1.3.4). The data points are converted to arrays using numpy (version 1.26.1). The data visualization task in the data analysis process uses the libraries matplotlib (version 3.4.3) and seaborn (version 0.11.2). Further data processing includes the feature extraction step (automatic and handcrafted), data splitting into training and testing, model training, hyper-parameter optimization and performance evaluation using evaluation metrics. All these steps have been carried out using the libraries of sklearn (version 0.24.2), XGBoost (version 1.5.1), catboost (version 1.1.1) and lightgbm (version 4.6.0).

5.2 Simulation analysis

The implementation part of the study utilizes various ML models and feature extraction methods that require the setting of various hyper-parameters. The feature extraction methods such as count vectorizer and TF-IDF vectorizer utilize the default hyper-parameters of the functions from sklearn. The ML LR model utilizes the default parameters that include the maximum number of iterations as 100; the

Table 2 Comparison of performance metrics with handcrafted and automatic feature extraction

Models	Feature Extraction	Average Accuracy (%)	Standard Deviation	Precision (%)	Recall (%)	F1 score (%)
LR	Handcrafted	99.38	0.016	99.33	98.79	99.06
	Automatic	99.48	0.019	99.99	97.79	98.88
Perceptron	Handcrafted	99.20	0.12	99.3	98.75	99.02
	Automatic	–	–	–	–	–
DT	Handcrafted	99.25	0.042	98.62	98.99	98.81
	Automatic	99.82	0.007	99.73	99.52	99.62
RF	**Handcrafted**	**99.57**	0.013	99.62	99.10	99.36
	Automatic	**99.82**	0.013	99.84	99.44	99.64
GB	Handcrafted	99.55	0.017	99.60	99.04	99.32
	Automatic	–	–	–	–	–
AdaBoost	Handcrafted	99.38	0.026	99.30	98.88	99.09
	Automatic	–	–	–	–	–
XGBoost	Handcrafted	99.55	0.022	99.55	99.13	99.34
	Automatic	99.79	0.007	99.78	99.37	99.57
KNN	Handcrafted	99.08	0.043	99.37	98.70	99.03
	Automatic	96.28	0.061	95.67	88.27	91.82
SVM	Handcrafted	99.40	0.032	99.37	98.70	99.03
	Automatic	–	–	–	–	–
CB	Handcrafted	99.56	0.021	99.65	99.02	99.34
	Automatic	99.77	0.010	99.64	99.30	99.47
MNB	Handcrafted	–	–	–	–	–
	Automatic	98.25	0.030	99.70	95.95	97.79
BNB	Handcrafted	–	–	–	–	–
	Automatic	99.64	0.012	99.97	98.45	99.20
LGBM	Handcrafted	99.54	0.013	99.51	99.05	99.275
	Automatic	99.78	0.011	99.72	99.34	99.53
PAC	Handcrafted	98.46	0.923	98.54	96.59	97.55
	Automatic	99.81	0.008	99.66	99.54	99.60

penalty is l2, which dictates the minimization of the sum of squares of the error. It has a default tolerance set to 0.0001. The perceptron model has the random state set to 42 with the rest being the default parameters, including alpha as 0.0001, l1 ratio as 0.15, maximum iteration 1000, tolerance of 0.001, and eta0 (learning rate) as 1. The DT classifier uses the default hyper-parameters where the criterion is the Gini index, the minimum samples split is 2, and the minimum samples per leaf is 1. The RF classifier has the number of estimators set to 600, while the other hyper-parameters are the same as the DT classifier.

The GB classifier has the number of estimators set to 700. At the same time, the other parameters are the default, which involves the loss as deviance, learning rate as 0.1, criterion as Friedman mean squared error minimum sample split 2, minimum sample per leaf 1, maximum depth of the tree three and tolerance being 0.0001. The AdaBoost classifier uses several estimators of 700, a learning rate of 0.1. In the case of XGBoost, the number of estimators is 700, while the other parameters are the same. K nearest neighbours have the hyper-parameter of several neighbours set to the nearest odd integer to the square root of the number of data points in the dataset. The weight is the distance. The rest of the parameters, such as metric, leaf size and p, have been kept as the default parameters, with Minkowski, 30 and 2 values, respectively. SVM classifier utilizes all the default parameters, including C as 1, the kernel is the radial basis function, degree 3 and tolerance 0.001. Lastly, in the case of the CB classifier, the number of estimators is set to 1000 while the rest of the parameters are kept null.

The hyper-parameter of the ML models where the feature extraction has been done using TF-IDF vectorizer is different from that of the handcrafted feature extraction. LR and DT classifiers utilize all the default parameters, thus the same as the handcrafted feature extraction. The XGBoost classifier has the number of estimators set to 700 while the rest of the parameters are the default once. The CB classifier has the number of estimators set to 500, while the rest is the default. The RF classifier has several estimators of 50, while the rest are the default. In the case of the k nearest neighbours classifier, the hyper-parameters are set as they have been set in the case of the handcrafted feature extraction. The MNB and BNB have been utilized for prediction purposes. Both the models use the default hyper-parameters where alpha is one and learning of prior probabilities being true.

For the ensemble methods like RF, the models have a high dependency on the number of trees rather than on the rest of the hyper-parameters, therefore, major focus has been given on the number of trees and the optimum value has been selected using the k-fold cross-validation. As a result, the best value for the number of trees turn out to be 600 thus making the RF model more robust. Additionally, the rest of the hyper-parameters have been kept as the default values since they don't add a negligible effect to the output of the RF model.

5.3 Evaluation metrics

In all the ML models, As performance measurements, assessment criteria such as accuracy, precision, recall, and F1 score were employed.

$$Accuracy = \frac{\zeta + \delta}{\zeta + \varepsilon + \delta + \eta} \tag{21}$$

$$Precision = \frac{\zeta}{\zeta + \varepsilon} \tag{22}$$

$$Recall = \frac{\zeta}{\zeta + \eta} \tag{23}$$

$$F1score = \frac{\zeta}{\zeta + \frac{1}{2}(\varepsilon + \eta)} \tag{24}$$

where, $\zeta = TruePositive$, $\delta = TrueNegative$, $\varepsilon = FalsePositive$, $\eta = FalseNegative$.

Accurate positive samples are those that are malicious, true negative samples are benign, false positive samples are falsely classified as malicious, and false negative samples are those that are malicious according to the data despite being classified as benign. F1 score indicates accuracy. Accuracy is the percentage of URLs correctly classified. Precision is the proportion of adequately identified positive samples to all positive predictions. The Recall is the ratio of the number of samples classified as malicious sample URLs classified as malicious to the total number of malicious samples. Low precision shows that the model classifies the URLs as malicious even if they are not. Low Recall shows that the model is not predicting the URLs correctly.

5.4 Performance evaluation

The models' prediction accuracy and tuning hyperparameter to obtain the best accuracy have been analysed. The optimum hyper-parameter has been selected using the k-fold cross validation with the number of trees being the hyper-parameter tuned to get the optimum value. Multiple values have been passed in for the number of trees and out of which 600 is the best value while keeping the rest of the hyper-parameters as default since not much significant change is visible. Therefore, the optimum value is used for the hyper-parameter using the k-fold cross-validation. The rest of the hyper-parameters haven't been changed since the classifier reached the optimum value with the default parameters itself. Thus, elimination the need for tuning any more hyper-parameters other than the number of trees. In addition to that, the major impact in RF is created by the number of trees since it is an ensemble method. The Table 2 shows the performance of each classification model and Fig. 4b, 5b, 6b and 7b is the graphical representation for the same. The RF classifier has outperformed all other classifiers with an accuracy of 99.57% and an F1 score of 99.36% as it can be seen Figs. 4b and 5b. All the models have performed well

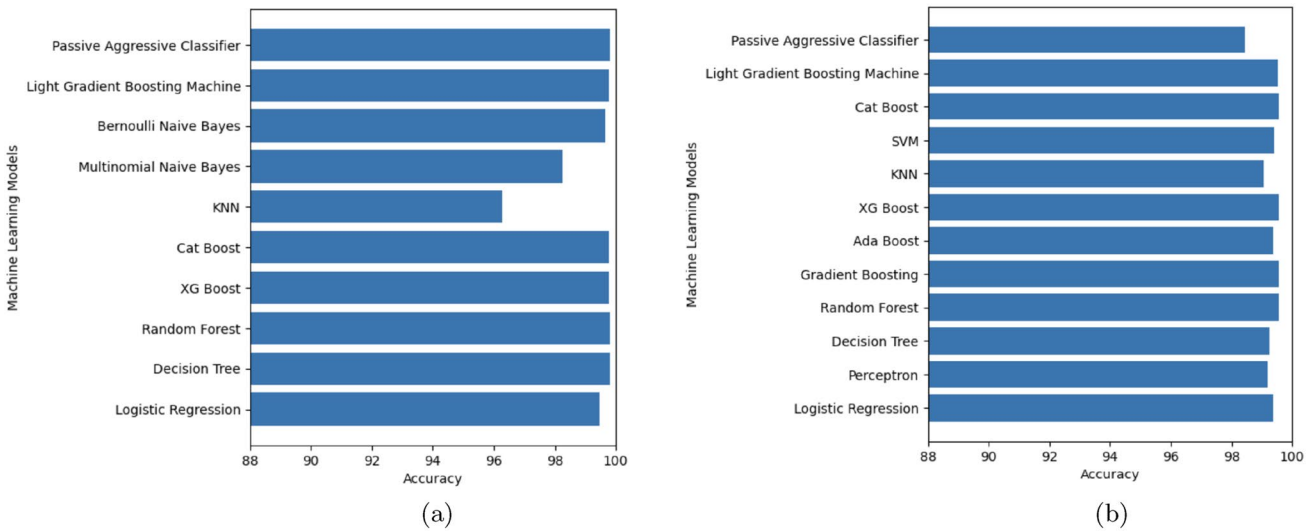

Fig. 4 Comparison of accuracy metric for (**a**) automatic feature extraction and (**b**) handcrafted feature extraction

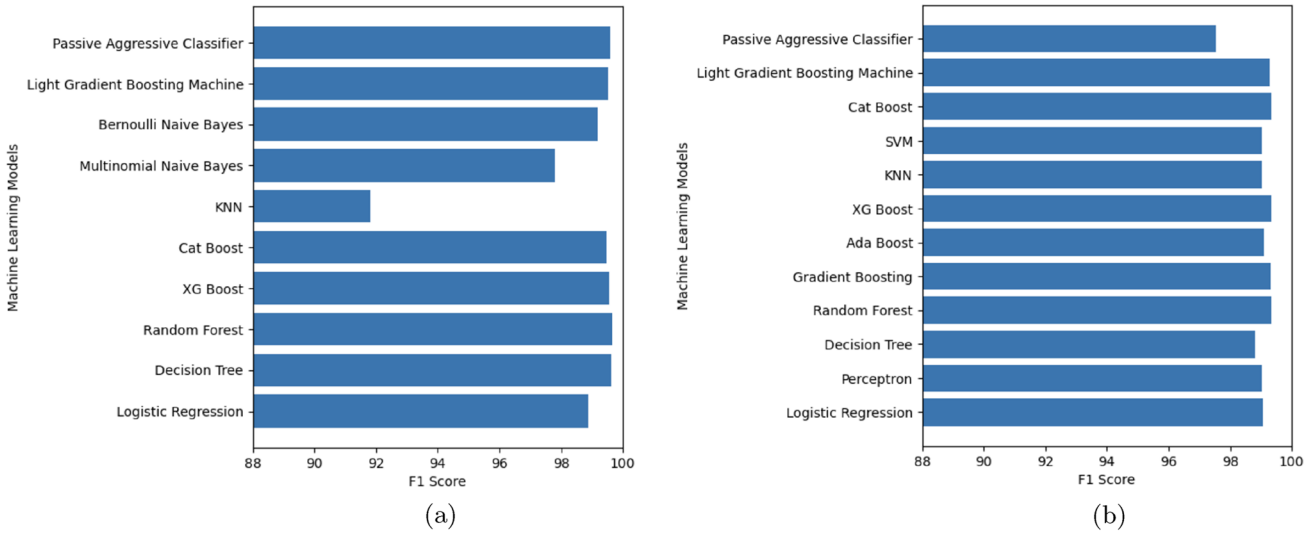

Fig. 5 Comparison of F1 score metric for (**a**) automatic feature extraction and (**b**) handcrafted feature extraction

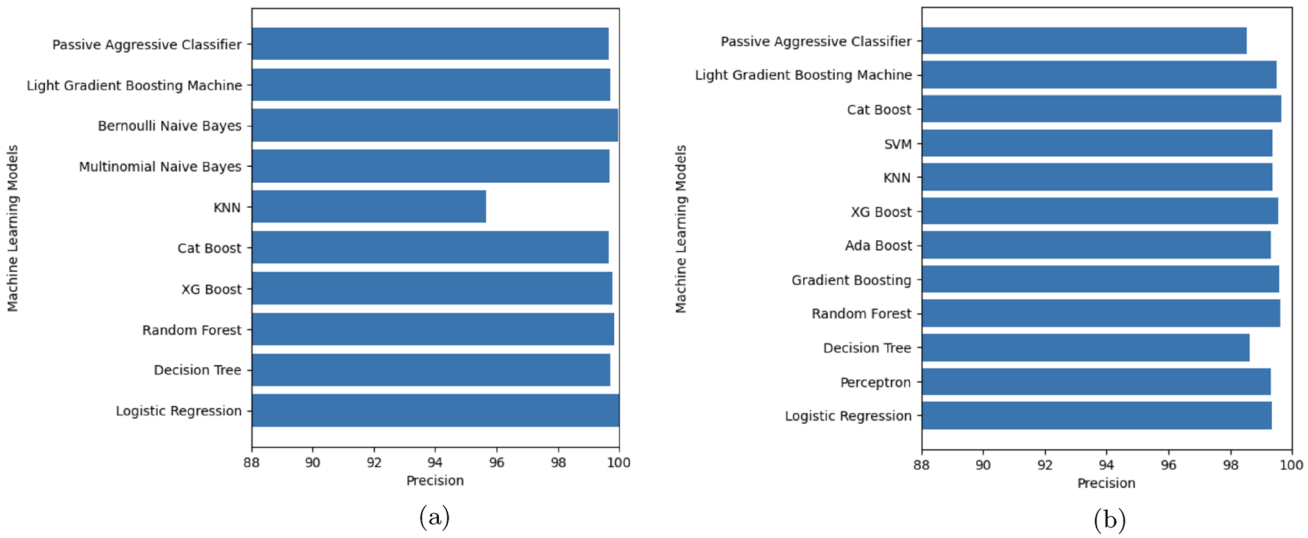

Fig. 6 Comparison of precision metric for (**a**) automatic feature extraction and (**b**) handcrafted feature extraction

with an accuracy above 99% and F1 score above 98.8%. K Nearest Neighbors is the least considered model here, with the lowest accuracy of 99.08% among all other models used. The RF classifier has the highest precision score of 99.62%, which shows that RF gives the most relevant results. XGBoost has the highest recall score of 99.13%, which shows that XGBoost classifies the relevant results correctly. Figure 8 shows the area under the receiver operating curve (ROC) of RF and KNN, respectively, the highest and lowest accuracy classifiers. This plot indicates the precision with which each model separates malicious and benign URLs at various classification thresholds. The x-axis is the False Positive Rate, the proportion of benign URLs classified as malicious, and the y-axis is the True Positive Rate, the proportion of correctly identified malicious URLs. Threshold selection critically impacts classification performance. Low thresholds often result in poor accuracy due to excessive false positives, while high thresholds generally yield better accuracy, precision, recall, and F1 scores. Our implementation employs a high threshold setting, which enables the model to maintain excellent performance metrics. All the classifiers used for the malicious URL detection have performed well, as AUC is almost the same for all the classifiers. Still, the RF model (AUC=0.9946) used whose line is closer to the top-left corner, has a better performance than all the classifiers. This is preferable in healthcare uses, where failing to detect harmful URLs and falsely blocking genuine healthcare resources can lead to serious problems. RF's steep spike near the origin suggests that it can be highly sensitive with low false positives, which

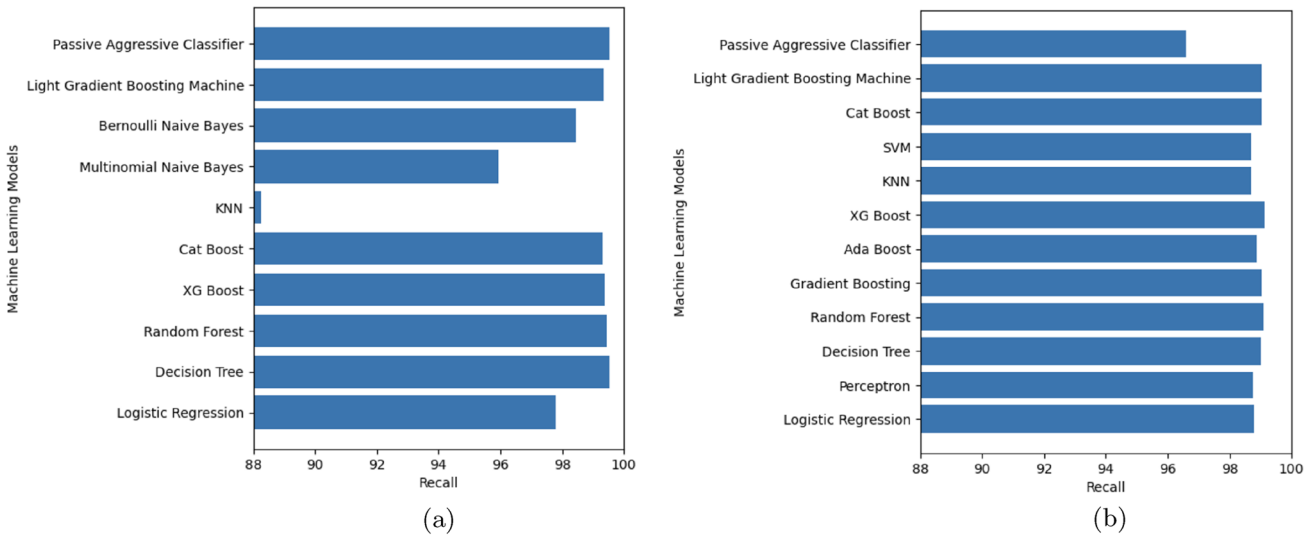

Fig. 7 Comparison of recall metric for (**a**) automatic feature extraction and (**b**) handcrafted feature extraction

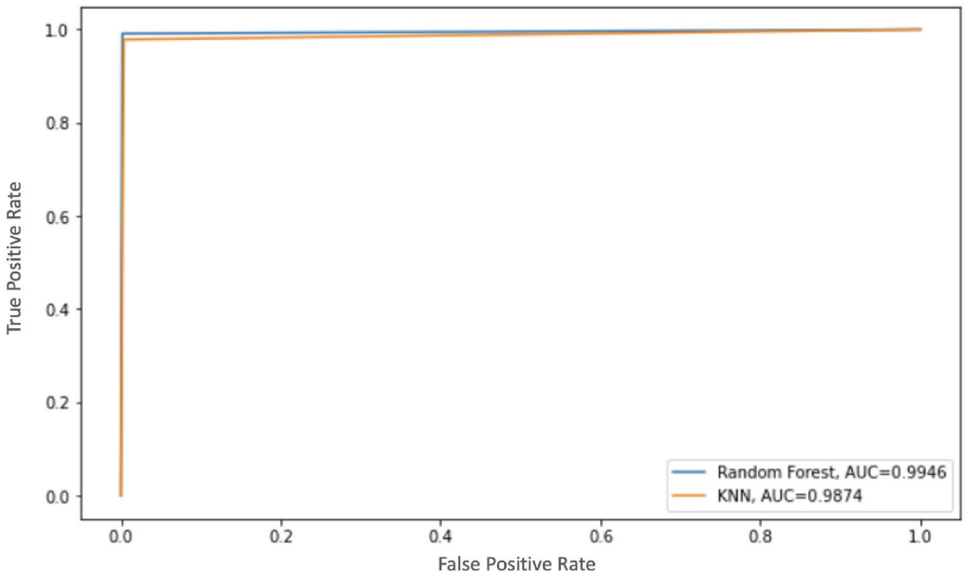

Fig. 8 ROC curve of RF and KNN—comparison of classification performance based on true positive rate vs. false positive rate

is most suitable for real-world applications. Table 4 and Fig. 9a represent the confusion matrix for automatic feature extraction for RF classifier. In addition to that, Table 3 and Fig. 9b shows the confusion matrix for handcrafted feature extraction for RF classifier with a clear indication that automatic feature extraction performs much better as compared to hand-crafted feature extraction since the former is able to cover the features that might necessarily be included in the latter method.

LR could not perform as well as RF because it cannot work well with non-linear decision boundaries as it does with linear boundaries. Thus, the dependent and independent variables are not linearly related. Even the perceptron mainly deals with linear decision boundaries. Therefore, it cannot perform better than a RF. The DT has more than ten levels for the given classification scenario. It gave a training accuracy of 100% and a testing accuracy of 99.25%. Thus, it can be concluded that the DT classifier has been overfitting the data. Even though the over-fitting is resolved through pruning, it caused a reduction in training and testing accuracy. The over-fitting problem, as in the case of DT classifiers, has been overcome by RF as it builds multiple DTs and employs a boot-strapping algorithm. More than one feature can act as the root node for the different DTs, such as 'https in URL,' 'WWW Present,' 'Digit to Alphabet ratio,' etc. This ensures having multiple different conditions in the root node rather than just a single root node, as in the case of the DT classifiers. Hence, the RF gives the best output. GB, LGBM and XGBoost perform nearly as well as RF, not just in case of accuracy but concerning other evaluation metrics; which

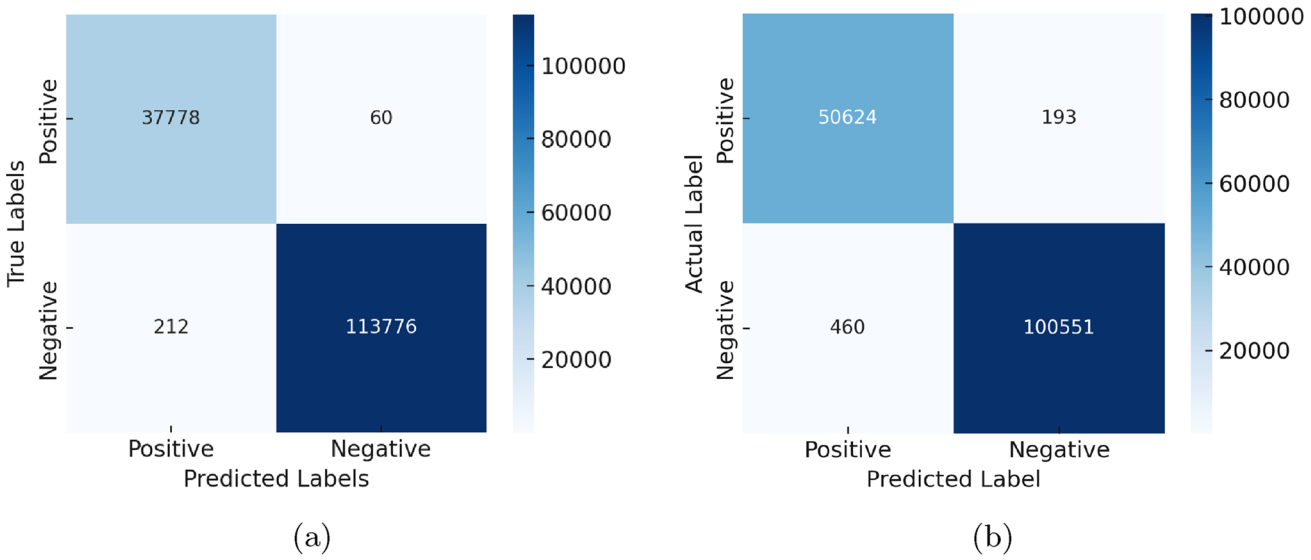

Fig. 9 Confusion matrix for (**a**) automatic and (**b**) handcrafted feature extraction with annotated TP, FP, TN, and FN

Table 3 Confusion matrix For RF classifier with handcrafted feature extraction

	Predicted Positive	Predicted Negative
Truely Positive	50624	193
Truely Negative	460	100551

Table 4 Confusion matrix For RF classifier with automatic feature extraction

	Predicted Positive	Predicted Negative
Truely Positive	37778	60
Truely Negative	212	113776

Table 5 Comparison of performance metrics with handcrafted and automatic feature extraction on different datasets for RF classifier

Datasets	Feature Extraction	Average Accuracy (%)	Standard Deviation
Original Dataset [26]	Handcrafted	99.57	0.013
	Automatic	99.82	0.013
Dataset-2 [37]	Handcrafted	99.35	0.029
	Automatic	99.73	0.023
Dataset-3 [38]	Handcrafted	95.68	0.069
	Automatic	99.51	0.022

makes it clear that ensemble methods are the key to solving the issue of detecting malicious URL through. While there is a level-wise growth (grows all leaves at the same level before proceeding to the next) in case of GB a leaf-wise growth (grows the tree by expanding the leaf that minimizes the loss the most) and in case of LGBM, both perform well and have overcome the issue of overfitting as visible in case of DT. Ada-boost penalises only those trees with a predicted output different from the true output. This reduces the overall updates, making them not converge to a greater extent. Hence, Ada-boost does not perform up to the mark. The KNN model is too simple for this classification problem, resulting in the lowest accuracy compared to other models. This indicates underfitting, as the model's simplicity prevents it from capturing the complexity of the dataset effectively. In the case of SVM, the radial kernel works far better than the linear or polynomial kernel because the dependent and independent variables are not linearly related. Referencing PAC, the lower accuracy as compared to the RF model is attributed to the linearity of the PAC model.

The TF-IDF method is performing well with the prospects of accuracy with a DT classifier, XGBoost, CB, and BNB. However, the DT classifier gets overfitted because it is just a single tree, while tree pruning causes the classifier's accuracy to reduce. In the case of XGBoost, multiple trees and residual addition are involved and gives good accuracy and a bias-variance trade-off. Moreover, a similar case is the CB classifier and LGBM classifier is visible. The BNB classifier uses the Bernoulli distribution to classify the data points based on multiple variables obtained by the TF-IDF vectorizer method. The performance of the TF-IDF vectorizer along with the different ML models is clearly visible by looking at Figs. 4a, 5a, 6a and 7a. Moreover, as the TF-IDF method finds out the term frequencies of the dataset, BNB performs better. This method gives little accuracy with LR, KNN, and MNB as they are more straightforward. PAC is able to perform very well despite it's model linearity because the concept that it operates is hinge loss which is a similar concept used in SVM. This adds high value to the PAC classifier thus making it highly comparable to the RF classifier in case of automatic feature extraction. LR and KNN are not well-known for use with text processing methods. Though MNB is used in the case of text processing, it

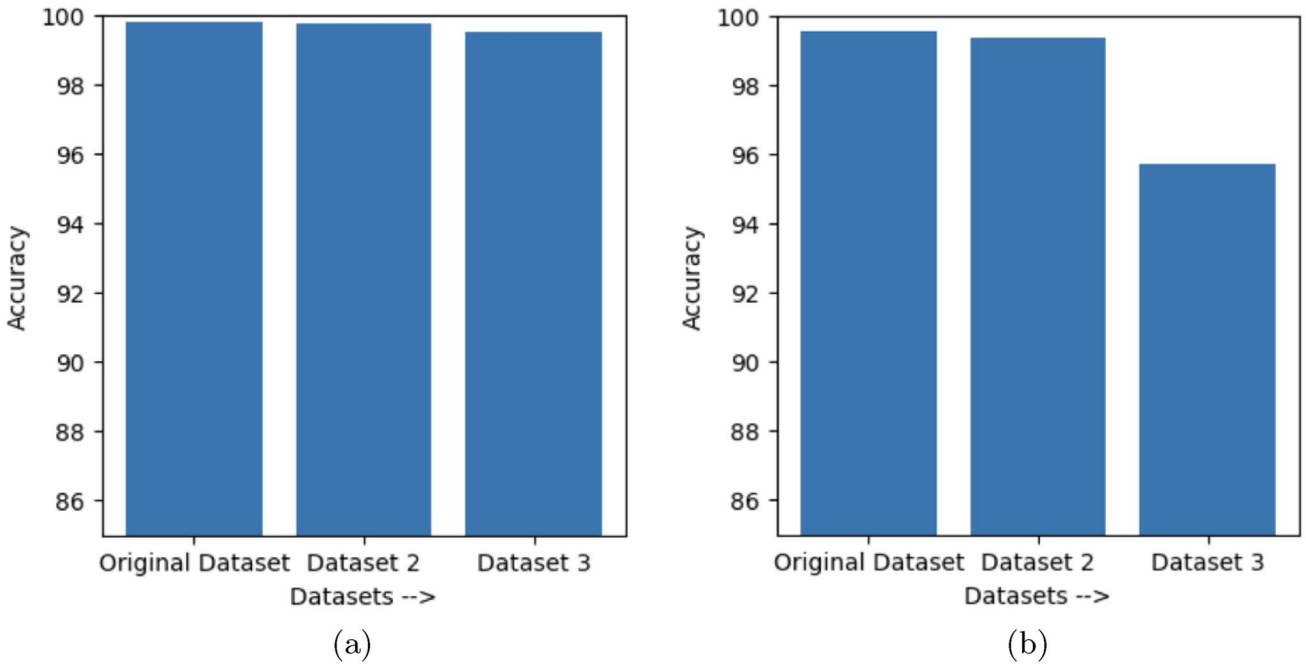

Fig. 10 Comparison of multiple datasets for RF classifier for (**a**) automatic and (**b**) handcrafted feature extraction

does not give a good accuracy due to its simplicity. The Area Under Curve of the ROC (AUC-ROC) for RF and KNN illustrates the difference in the confidence levels of the two models. The ROC curve for RF is steeper and stays closer to the top-left corner compared to KNN, indicating a more confident classification. This higher confidence is attributed to the superior training and generalization capabilities of the RF model.

Further, comparison has been carried out using multiple open source datasets for malicious URLs and the results for the same are presented in Table 5 using RF. Although the hand-crafted feature extraction isn't working as good for Dataset-3, RF is showing similar results for all the datasets when pre-processing is carried out using automatic feature extraction. Moreover, the Fig. 10a and Fig. 10b represents the graphical comparisons of the average accuracies for the RF classifier implemented on each of the datasets given in Table 5. It shows the robustness of the RF model when combined with the automatic feature extraction (TF-IDF) method such that the model is able to perform just as good for every dataset rather than doing well at just a single set of data points. The standard deviation in Tables 2 and 5 is used to indicate the low variation in the results of the given models. The lower the standard deviation, the more stable the model in terms of giving accuracy and the better for usage since it performs well in the predictable range. Thus, indicating the predictability of the models and reliability over the enlisted classifiers. The less the standard deviation, the more reliable the classification model.

6 Conclusion

In our work, two feature extraction methodologies have been compared to discriminate between the malicious and non-malicious URLs. One is hand crafted feature extraction and the other one being automatic feature extraction. Among these, the automatic feature extraction method has proven to be better as compared to the hand crafted feature extraction. Although RF has performed the best in both the methods, the accuracy it shows in case of automatic feature extraction is 99.82% and that in case of hand crafted feature extraction is 99.57%. In matters of other metrics such as precision score, recall score and F1 score, the automatic feature extraction gives better results with the values being 99.84%, 99.44% and 99.64% respectively. Therefore, it is clear that automatic feature extraction overpowers hand crafted feature extraction in case of classification of URLs into malicious and non-malicious.

Future work involves the use of deep learning models including transformers and pre-trained deep learning models as well for the classification of malicious and non-malicious URLs. Additionally, there would an increase in the data such that the model can be more generalized and thus more common features can be extracted by the deep learning models. Moreover, we recognize the importance of optimizing IoT components for enhanced performance. Future enhancements will focus on: Implementing distributed ML frameworks to accelerate data processing for large-scale IoT networks, Reducing computational overhead by selecting

the most relevant features dynamically, minimizing the power consumption of IoT devices.

Author contributions Conceptualization, A.N. and T.V.; methodology, H.V.; software, P.P. and M.P; validation, A.N., T.V. and C.B.; formal analysis, H.V. and M.P.; investigation, L.C.; resources, G.D.N.; data curation, P.P.; writing—original draft preparation, P.P., M.P., and H.V.; writing—review and editing, A.N.; visualization, C.B.; supervision, G.D.N.; project administration, L.C.; funding acquisition, G.D.N. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Università del Salento within the CRUI-CARE Agreement. Publication fee were covered by a CRUI-CARE agreement

Data availability Data used in this work are available online as a public dataset.

Materials availability Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare no Conflict of interest.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Wang, L., Alexander, C.: Big data analytics in healthcare systems. Int. J. Math. Eng. Manage. Sci. **4**, 17–26 (2019). https://doi.org/10.33889/IJMEMS.2019.4.1-002
2. Dutta Pramanik, P., Pal, S., Mukherjee, M.: Healthcare big data: a comprehensive overview, 72–100 (2018). https://doi.org/10.4018/978-1-5225-7071-4.ch004
3. Jabali, A.K., Waris, A., Khan, D.I., Ahmed, S., Hourani, R.J.: Electronic health records: three decades of bibliometric research productivity analysis and some insights. Inform. Med. Unlocked **29**, 100872 (2022). https://doi.org/10.1016/j.imu.2022.100872
4. Upadhyay, S., Hu, H.-f.: A qualitative analysis of the impact of electronic health records (ehr) on healthcare quality and safety: Clinicians' lived experiences. Health Services Insights 15, 11786329211070722 (2022) https://doi.org/10.1177/11786329211070722. PMID: 35273449
5. Keshta, I., Odeh, A.: Security and privacy of electronic health records: concerns and challenges. Egypt. Inform. J. **22**(2), 177–183 (2021). https://doi.org/10.1016/j.eij.2020.07.003
6. Sahi, A., Lai, D., Li, Y.: A review of the state of the art in privacy and security in the ehealth cloud. IEEE Access **9**, 104127–104141 (2021). https://doi.org/10.1109/ACCESS.2021.3098708
7. Yüksel, B., Küpçü, A., Ozkasap, O.: Research issues for privacy and security of electronic health services. Future Generat. Comput. Syst. (2016). https://doi.org/10.1016/j.future.2016.08.011
8. Alkanhel, R., El-kenawy, E.-S.M., A. Abdelhamid, A., Ibrahim, A., Alohali, M.A., Abotaleb, M., Khafaga, D.S.: Network intrusion detection based on feature selection and hybrid metaheuristic optimization. Computers, Materials & Continua 74(2), 2677–2693 (2023) https://doi.org/10.32604/cmc.2023.033273
9. Alsaedi, M., Ghaleb, F.A., Saeed, F., Ahmad, J., Alasli, M.: Cyber threat intelligence-based malicious url detection model using ensemble learning. Sensors (2022). https://doi.org/10.3390/s22093373
10. ALfouzan, N.A., C, N.: A systematic approach for malware url recognition. In: 2022 2nd international conference on computing and information technology (ICCIT), pp. 325–329 (2022). https://doi.org/10.1109/ICCIT52419.2022.9711614
11. Sánchez-Paniagua, M., Fernández, E.F., Alegre, E., Al-Nabki, W., González-Castro, V.: Phishing url detection: a real-case scenario through login urls. IEEE Access **10**, 42949–42960 (2022). https://doi.org/10.1109/ACCESS.2022.3168681
12. Liang, Y., Wang, Q., Xiong, K., Zheng, X., Yu, Z., Zeng, D.: Robust detection of malicious urls with self-paced wide & deep learning. IEEE Trans. Dependable Secure Comput. **19**(2), 717–730 (2022). https://doi.org/10.1109/TDSC.2021.3121388
13. Yuan, J., Chen, G., Tian, S., Pei, X.: Malicious url detection based on a parallel neural joint model. IEEE Access **9**, 9464–9472 (2021). https://doi.org/10.1109/ACCESS.2021.3049625
14. Wejinya, G., Bhatia, S.: Machine Learning for Malicious URL Detection, 463–472 (2021). https://doi.org/10.1007/978-981-15-8289-9_45
15. Yunusa, A.M.: Detection of malicious url using machine learning: a review (2023) https://doi.org/10.13140/RG.2.2.11778.86729
16. Lam, N.: Developing a framework for detecting phishing urls using machine learning. Int. J. Emerg. Technol. Adv. Eng. **11**, 61–67 (2021). https://doi.org/10.46338/ijetae1121_08
17. El-kenawy, E.-S.M., Khodadadi, N., Mirjalili, S., Abdelhamid, A.A., Eid, M.M., Ibrahim, A.: Greylag goose optimization: nature-inspired optimization algorithm. Expert Syst. Appl. **238**, 122147 (2024). https://doi.org/10.1016/j.eswa.2023.122147
18. Alsaedi, M., Ghaleb, F.A., Saeed, F., Ahmad, J., Alasli, M.: Cyber threat intelligence-based malicious url detection model using ensemble learning. Sensors **22**(9) (2022)
19. Chiramdasu, R., Srivastava, G., Bhattacharya, S., Reddy, P.K., Reddy Gadekallu, T.: Malicious url detection using logistic regression. In: 2021 IEEE International conference on Omni-Layer intelligent systems (COINS), pp. 1–6 (2021). https://doi.org/10.1109/COINS51742.2021.9524269
20. Wagan, A.A., Li, Q., Zaland, Z., Marjan, S., Bozdar, D.K., Hussain, A., Mirza, A.M., Baryalai, M.: A unified learning approach for malicious domain name detection. Axioms (2023). https://doi.org/10.3390/axioms12050458
21. Gupta, B.B., Yadav, K., Razzak, I., Psannis, K., Castiglione, A., Chang, X.: A novel approach for phishing urls detection using lexical based machine learning in a real-time environment. Comput. Commun. **175**, 47–57 (2021). https://doi.org/10.1016/j.comcom.2021.04.023

22. Abad, S., Gholamy, H., Aslani, M.: Classification of malicious urls using machine learning. Sensors (2023). https://doi.org/10.3390/s23187760
23. Myriam, H., A. Abdelhamid, A., El-Kenawy, E.-S.M., Ibrahim, A., Eid, M.M., Jamjoom, M.M., Khafaga, D.S.: Advanced meta-heuristic algorithm based on particle swarm and al-biruni earth radius optimization methods for oral cancer detection. IEEE Access 11, 23681–23700 (2023) https://doi.org/10.1109/ACCESS.2023.3253430
24. Yan, X., Xu, Y., Cui, B., Zhang, S., Guo, T., Li, C.: Learning url embedding for malicious website detection. IEEE Trans. Industr. Inf. **16**(10), 6673–6681 (2020). https://doi.org/10.1109/TII.2020.2977886
25. Manyumwa, T., Chapita, P.F., Wu, H., Ji, S.: Towards fighting cybercrime: Malicious url attack type detection using multiclass classification. In: 2020 IEEE international conference on big data (Big Data), pp. 1813–1822 (2020). https://doi.org/10.1109/BigData50022.2020.9378029
26. Nowroozi, E., -, A., Mohammadi, M., Conti, M.: Pristine and malicious urls (2022) https://doi.org/10.21227/2ph5-xc09
27. Computer Science and Intelligent Systems Research Center, Blacksburg 24060, Virginia, USA, Khaled, K., Department of Interdisciplinary Courses in Engineering, Chitkara University Institute of Engineering Technology, Chitkara University, Punjab, India, Singla, M.K.: Predictive analysis of groundwater resources using random forest regression. Journal of Artificial Intelligence and Metaheuristics 09(01), 11–19 (2025)
28. Nowroozi, E., Abhishek, Mohammadi, M., Conti, M.: An adversarial attack analysis on malicious advertisement url detection framework. IEEE Transactions on Network and Service Management 20(2), 1332–1344 (2023) https://doi.org/10.1109/TNSM.2022.3225217
29. Maftoun, M., Shadkam, N., Komamardakhi, S.S.S., Mansor, Z., Joloudari, J.H.: Malicious URL Detection using optimized Hist Gradient Boosting Classifier based on grid search method (2024). arxiv:abs/2406.10286
30. Reyes-Dorta, N., Caballero-Gil, P., Rosa-Remedios, C.: Detection of malicious urls using machine learning. Springer (2024). https://doi.org/10.1007/s11276-024-03700-w
31. Abad, S., Gholamy, H., Aslani, M.: Classification of malicious urls using machine learning. Sensors (2023). https://doi.org/10.3390/s23187760
32. Rafsanjani, A.S., Kamaruddin, N.B., Rusli, H.M., Dabbagh, M.: Qsecr: secure qr code scanner according to a novel malicious url detection framework. IEEE Access **11**, 92523–92539 (2023). https://doi.org/10.1109/ACCESS.2023.3291811
33. He, S., Li, B., Peng, H., Xin, J., Zhang, E.: An effective cost-sensitive xgboost method for malicious urls detection in imbalanced dataset. IEEE Access **9**, 93089–93096 (2021). https://doi.org/10.1109/ACCESS.2021.3093094
34. Indrasiri, P.L., Halgamuge, M.N., Mohammad, A.: Robust ensemble machine learning model for filtering phishing urls: expandable random gradient stacked voting classifier (erg-svc). IEEE Access **9**, 150142–150161 (2021). https://doi.org/10.1109/ACCESS.2021.3124628
35. Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., Bindhumadhava, B.S.: Phishing website classification and detection using machine learning. In: 2020 international conference on computer communication and informatics (ICCCI), pp. 1–6 (2020). https://doi.org/10.1109/ICCCI48352.2020.9104161
36. Computer Science and Intelligent Systems Research Center, Blacksburg 24060, Virginia, USA, Mahmoud, M.: A review on waste management techniques for sustainable energy production. Metaheuristic Optimization Review 3(2), 47–58 (2025)
37. Prasad, A., Chandra, S.: PhiUSIIL Phishing URL (Website). UCI Machine Learning Repository. https://doi.org/10.1016/j.cose.2023.103545 (2024)
38. Malicious URLs dataset — kaggle.com. https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset. [Accessed 02-04-2025]
39. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)