An Approach to Detect Phishing Websites with Features Selection Method and Ensemble Learning

Mahmuda Khatun[1]
Department of CSE
Comilla University
Cumilla - 3506, Bangladesh

MD Akib Ikbal Mozumder[2]
Department of CSE
Comilla University
Cumilla - 3506, Bangladesh

Md. Nazmul Hasan Polash[3]
Department of CSE
Comilla University
Cumilla - 3506, Bangladesh

Md. Rakib Hasan[4]
Dept. of Information & Comm. Technology
Comilla University
Cumilla - 3506, Bangladesh

Khalil Ahammad[5]
Department of CSE
Comilla University
Cumilla - 3506, Bangladesh

MD. Shibly Shaiham[6]
Department of CSE
CUET
Chattagram-4349, Bangladesh

***Abstract*—Nowadays, phishing is a major problem on a global scale. Everyone must use the internet in today's society in order to cope up in the real world. As a result, internet crime like phishing has become a serious issue throughout the world. This type of crime can be committed by anyone; all they need is a computer. Additionally, hacking may now be learned quickly by anyone with programming and mathematical skills. The adoption of various techniques by anti-phishing toolbars, such as machine learning, may enable users to quickly identify a fake website. As a result, researchers are now particularly interested in the problem of detecting fraudulent websites. Machine learning techniques have been offered throughout the entire process to more precisely identify fraudulent websites. To find the best accurate outcome, classification with random parameter tuning and ensemble based approaches are utilized. A user-friendly interface has also been suggested to make the system more accessible to the public.**

Keywords*—*Machine learning; deep learning; catboost; LGBM; embedded; react-native; flask

I. Introduction

Technology has made individuals more dependent than they have ever been before. As the price of electronic devices such as smartphones, tablets, personal computers, laptops, and so on continues to drop, an ever-increasing number of people are able to purchase them and are using them. However, the rise of cyber dangers has been the most significant in recent decades. Phishing, which is responsible individually for 90 percent [1] of the data breaches, is one of the most common methods that people are tricked into giving over their personal information. People who conduct most of their financial transactions and shopping online are the most likely to fall victim to phishing scams. Criminals used to extort money from unsuspecting victims by threatening them with guns, stealing their cars, or by using force. When compared to less developed countries, industrialized nations have a lower incidence of this particular form of criminal activity. But in recent years, phishing has become a significant issue all around the world. This is due to the fact that one does not require any kind of weapon in order to commit this form of crime; all that is required is a computer. In addition, the availability of books and guides to follow on the internet makes it possible for anyone who is mathematically savvy and has some experience with programming to learn how to hack. The Federal Bureau of Investigation estimates that more than 26 billion dollars have been stolen from businesses and individuals around the world. In addition, the number of new websites that are used for phishing is continually expanding. Fig. 1 indicates new phishing websites are increasing year by year which is an immense threat for users. Therefore, cutting-edge research needs to be carried out in order to neutralize this danger. Also many cyber security training are providing by both government or by private companies which rises the awareness of people basically in the form of games [2] to [3].

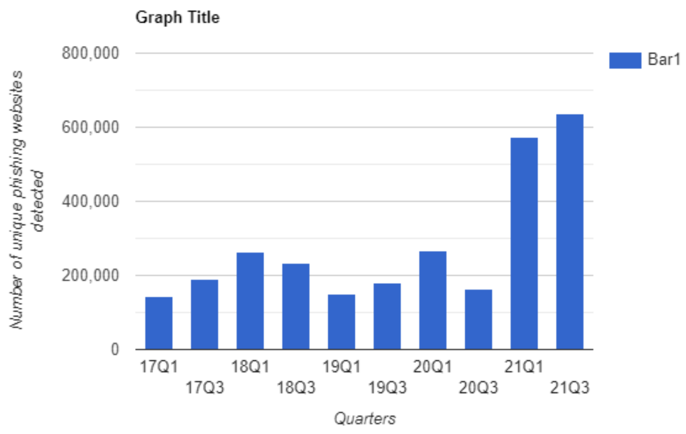


Fig. 1. The Number of New Phishing Websites from First Quarter of 2017 upto Third Quarter of 2021 by Statista.

Internet scams that use email, social media, and websites to steal private information are called phishing. Attackers take advantage of both system flaws and user ignorance. As a result, researchers must design solutions at both the technical and user-level levels. It is possible to protect yourself from Phishing by using a variety of different methods. Anti-phishing toolbars primarily protect users on their computers' local machines from phishing attacks. Games and other awareness campaigns are used to educate people about the issue [4] to[5].

Machine learning, rule-based, visual similarity, and other non-content-based methods are used by anti-phishing toolbars to identify phishing sites. Malicious URL detection technology aids users in spotting malicious links and wards off attacks from harmful websites. So, the first motivation is to give safety to the users. Another motivation is to increase the reliability of such technology. For that, increasing the accuracy of the system is a crucial factor which has been done in this research work. As a result, we conducted a comprehensive study of the current literature on phishing types, anti-phishing techniques, and anti-phishing toolbars in order to better understand the faults of existing toolbars and provide researchers with improved solutions. In order to compile the papers, we used Google Scholar as a database. Supervised Machine Learning with different Features Selections has been used in the past to detect phishing websites [6]. Classifiers that use wrapper-based techniques typically have the most useful features and the best overall performance [7] to [8]. Multi-model simulations are used in this phase for higher performance accuracy [9]. In order to achieve the highest level of accuracy, a hybrid model incorporates the greatest elements of multiple models while minimizing the negatives of each. The most accurate results are obtained using ensemble methods [10] to [11]. Using wrapper feature selection and ensemble learning, a phishing detection software will be created. Wrapping features selection approach delivers the most desired features in order to acquire the most accurate outcome. In addition, an ensemble technique is utilized to find the accuracy of all classifiers and select the best classifier to fit with it in order to forecast phishing websites.

Section II shows the literature review and Section III briefly describes the methodology and design of our study. Section IV elaborates the experimental results and Section V concludes with limitation and some recommendations of future work of the study.

II. Literature Review

Many researchers worked previously on analysing phishing websites. Our technique is to combine their works and get a better result. Also there exists many phishing apps and some are in the form of games which literally use live servers to detect urls [12]. We went through the past works and try to improve the accuracy of models and propose a user friendly interface to detect them.If we want to run a detection app successfully then we have to dig into the process first. M.A Tahir [11] proposed a hybrid model to detect phishing sites using supervised learning algorithm. He used machine learning methods and combine them to find the best accuracy but skip the feature selection methods. Zamir [13] proposed a diverse machine learning method to detect phishing websites wheres they skip the part of using feature extraction methods. Sharivari [14] also proposed a system where they used almost every classification techniques. R.kohavi [7] shows wrapper based features selection classifier work with all the possible features subsets and utilizes a ml classifier as an evaluation function of the features subsets. The highest evaluation is considered as the best feature subset. W.Ali [6] tried to propose a system with machine learning classifier and the feature selection method. He found that random forest with wrapper based feature selection method gives the highest accuracy of 97.3%. The most accurated result found is 97.4%. Our proposed system is to improve their accuracy by using same dataset. Also there are so many apps or games and they mainly use live servers or open source to detect urls [12].Our project is to make a user friendly app with our proposed ml method.

There exists many methods to detect fake urls. Some based on machine learning techniques and skip features selection methods. In our project we use classifiers as well as feature selection methods and also implement this as an application. The researchers applied simple feature identification and extraction techniques. It can also be possible to get more accurate results by using more powerful classifiers in ML such as category boosting and Deep Learning algorithms also can handle this type of problem. The existing models did not extract as much as features like us. So, the reduced number of feature extraction in the existing models is a gap. There is scope to improve the accuracy of the existing models. In the next section we will describe how we overcome this lacking found a better way to overcome some problems.

III. Methodology and Design

Methodology is considered as the best part of any research. It implements the answer of the questions start with “How?”. It’s a tremendous way to ensure the acceptability of the work through valid and reliable results. For this proposed method, we have gone through several steps. At first valid data have been collected. Then this data have gone through several steps till a user interface use this model to predict the urls. Machine learning classifiers combining with feature selection methods have been used and react native is used for given it a real life application.

There are two major steps to illustrate this project. First of all selecting the best features among all features and then select an ensemble machine learning method to get the result.

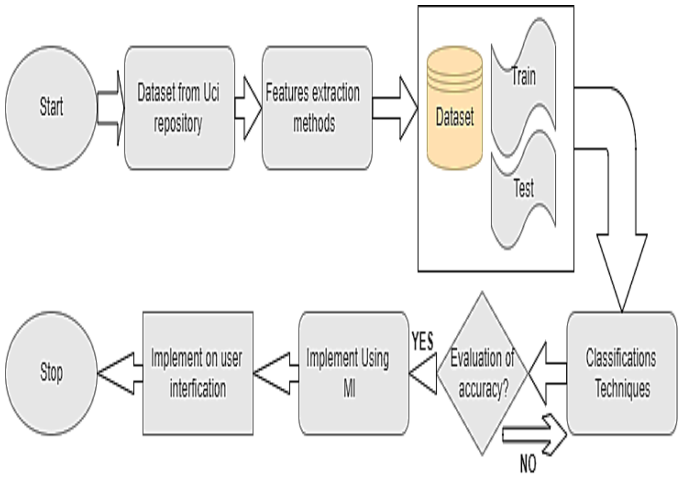

Start
Dataset from Uci repository
Features extraction methods
Dataset
Train
Test
Classifications Techniques
Evaluation of accuracy?
YES
NO
Implement Using Ml
Implement on user interfication
Stop

Fig. 2. Flow Chart for whole Process.

Fig. 2 illustrates the methodology of phishing website detection based on supervised machine learning classifiers with features selection method.

Fig. 3 illustrates the classes we have used for Ml model classification and feature extraction methods.They have been rotating to find the best accuracy among them.

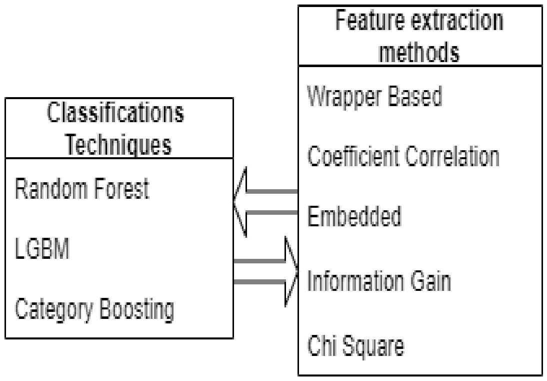


Fig. 3. Loop between Ml Model Classifiers and Features Extraction Technique.

Five steps are required to be accomplished in order to detect the phishing website: dataset collection, features extraction, features selection, training of machine learning classifiers, and evaluation of machine learning classifiers

The dataset of our project were collected from the UCI Machine Learning Repository, which is freely available for use. This dataset consists of 11055 rows and 30 columns which were used to extract several website features [15]

A. Explanation of Proposed Method

The complete explanation of the proposed method is given in this section. The basics of all procedures in the methodologies are discussed.

1) Data Preprocessing and Analysing: For building a good accurate model preprocessing of data is must. Otherwise the model may fail to give correct results. Data processing is a term which is basically processing of raw unusable data to suitable machine data.Using J48 algorithm [16] is very helpful in examine the data categorically and continuously. Fig. 4 shows the preprocessing procedure of the data. Here we have used basic types of data preprocessing methods in our project.

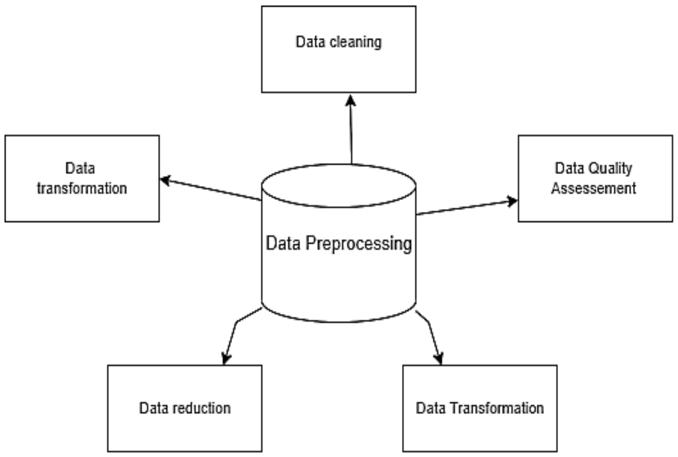


Fig. 4. Data Pre-Processing.

The following are some fundamental data preprocessing methodologies that have been implemented in the proposed method.

1) We have dropped one column which only indicates the index value sequentially for all the records.
2) Then We have analysis the unique classes for each of the data.
3) Next we have replaced all the values of records to 0 or 1
4) Those records who have missing values are also eliminated.

2) Feature Extraction: ML can't work best on choosing all the features. So feature selection method is used for dealing with crude content straightforwardly. It reduces the dimension of feature space.During feature extraction, uncorrelated and useless features will be removed which can help to fit the best useful algorithm and can improve the accuracy of the model

Wrapper Based: Subset of features are used and with each subset, train a model. We add or remove features based on the accuracy given by the subsets.It is also possible to utilize wrapper classification algorithms that combine dimensionality reduction and classification, although their computational costs are considerable and their discriminative power is limited. Furthermore, in order to achieve high accuracy, these strategies rely on the effective selection of classifiers. There are three types of wrapper based selection:

1) Forward Selection
2) Backward Elimination
3) Stepwise Selection

In our proposed method Stepwise selection secured best accuracy among all of the features selection methods.

Correlation Based: High correlated features depends each other and linearly inter dependent, hence have almost the same effect.When a feature expansion doesn't result in an improvement, the algorithm moves on to the next best unexpanded subset. The entire feature subset space is searched by this approach without any restrictions. As a result, there should be a limit on backtracking. The program then returns the feature subset that produced the highest merit up to that point.

Information Gain: The information gained is a system where amount of information improved before splitting them are counted. This is actually the mutual information between two random variables.Determining an attribute's relevancy and, consequently, its position inside the decision-tree, is the main goal of the Information Gain. An attribute (variable) with numerous distinct values prevents the information gain from effectively differentiating among the attributes.

Chi Square: Categories in a dataset are tested using the chi-square method. We determine the Chi-square among each element and the intended outcome and then choose the features with the highest Chi-square scores. It assesses whether the relationship between two categorical data in the sample corresponds to their true relationship in the population.

3) Machine Learning Algorithm: Machine learning is a learning technique and getting new information without explicit instruction with the aid of training data. Basically, there are two different categories of machine learning methods.

1) Supervised

2) Unsupervised

The method of learning from leveled data known as "supervised learning" provides an output for each input. In accordance with this, the algorithms operate. Unsupervised learning refers to the type of learning when the training or operation is carried out on an unlabeled dataset. To discover related features, the algorithm aggregates data of a similar type. Whatever the case, the labeled data is used to develop our model. A supervised learning problem can thus be used to frame the issue. There are two different kinds of supervised learning algorithms.

1) Classification
2) Regression

As we have used Supervised learning technique and dataset is classifed, in the next part we will only discuss about supervised classification methods.

Random Forest: A random forest is a ml method for tackling classification and regression issues. It makes use of ensemble learning, a method for solving complicated issues by combining a number of classifiers.

Numerous decision trees make up a random forest algorithm. The random forest algorithm creates a "forest" that is trained via bagging or bootstrap aggregation. The accuracy of ml algorithms is increased by bagging, an ensemble meta-algorithm.

Based on the predictions made by the decision trees, the (random forest) algorithm determines the outcome. It makes predictions by averaging or averaging out the results from different trees. The accuracy of the result grows as the number of trees increases.

Light Gradient Boosted Machine: A framework and a type of gradient boosting is called a light gradient boosting machine, or LGBM. Light GBM is based on Decision tree methods, just as another gradient boosting method. We can decrease memory utilization and boost efficiency with the aid of Light GBM.

The primary distinction between Light GBM and other gradient boosting frameworks is that Light GBM grows leaf-wise and in a vertical orientation. The other algorithms, however, grow horizontally in a level-wise fashion. The leaf that produces the least inaccuracy and the most efficiency is chosen by Light GBM. This approach is significantly more beneficial in lowering the mistake percentage. In other words, it expands leaf-wise whilst others grow level-wise.

Category Boosting: The CatBoost algorithm plays an essential part for supervised machine learning applications and is based on Gradient Descent. It will work effectively for issues involving categorical data. A series of decision trees are generated sequentially when training this model because the CatBoost technique is based on gradient decision trees. Each new tree that is created as training goes on has a lower loss than the one before it. It is employed for a variety of functions, including weather forecasting, self-driving cars, recommendation systems, personal assistants, and search.

4) React Native APP: Basics of react native: JavaScript provides the framework for React Native, which allows developers to write genuine mobile applications that run natively on iOS and Android devices [17]. It is based on React, which is a JavaScript toolkit that Facebook uses for designing user interfaces; however, rather than targeting browsers, it targets mobile platforms. To put it another way, web developers now have the ability to create mobile applications that have a look and feel that is genuinely "native", and they can do it from the convenience of a JavaScript library that we are already familiar with and adore. In addition, the majority of the code that you create may be shared between platforms, which makes it simple to develop applications for both Android and iOS at the same time using React Native. Fig. 5 indicates that how server and app are interacting with each other.

Architecture of our react native app:

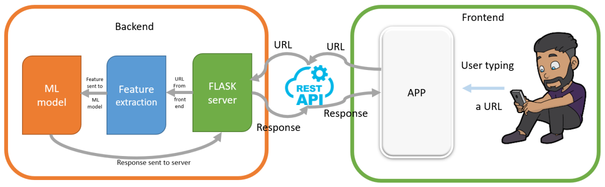


Fig. 5. Architecture of React Native.

The architecture has mainly two parts. Backend and frontend. Each part is divided into some essential parts.

5) Implementation of Machine Learning Model:: The steps required to implement are as follows:

- **Step-1:** At first we collected the data from online [15]
- **Step-2:** Then we did some analysis of the data which we have already discussed.
- **Step-3:** We did some preprocessing of the data that is necessary for building the models. We first balance the dataset as the data were very class imbalanced. Then we classified data and discard the record which have null values
- **Step-4:** Then we did the most important part which is feature extraction. We have used the wrapper based,embedded method, correlation coefficient, information gain and chi square. Then we implemented this techniques one by one with ensemble machine learning techniques.
- **Step-5:** After feature extraction, we have split the dataset into train and test set. We have used 70split from the same dataset.
- **Step-6:** We have built the machine learning model. We have used ensemble models [10] like Random forest, LGBM, Catboosting method. And then We have implemented this methods with features extraction methods which We have discussed already in the previous part.
- **Step-7:** After building the model we evaluated the performance of the model. And then implemented this methods to the user interface.

6) Implementation of User Interface using our Machine Learning Model:: Here we will discuss about the user interface which will be a real life url detector using our machine learning approach [18]. Backened: Here a server is created to communicate with the frontend and ML model. It sends the response of ML model to the frontend.

Flask Server
Flask is a python framework. With the help of Flask, a server was created. It receives URL via REST API from frontend. It is also responsible to send the response of ML model to REST API.

Extracting Features
Features from URL are extracted at this end. The extracted features are then fed into the ML model.

Response form ML Model
With the help of the features, the ML model creates a response whether the URL is phishing or safe.

Frontend: The frontend supplies the URL to the Backend through REST API.

App
The react native app receives input from user. The URL is saved to its state and then it is sent to the REST API. The API sends it to the server.

User
The user gives an URL as input to the App. The user also can see the response of the ML model in the app interface.

IV. Results

Performance are the outcomes and the most important parts of a project. The better a model performs the more useful that model is. In the previous chapters all the steps for the proposed methodology has been explained. Now it's time to evaluate the model. In this section we will describe the overall outcomes and we will show our proposed model with the traditional approaches. The performance and the real life user interface will be discussed here.

A. Dataset Description

The dataset of our project have already discussed in the previous chapter. Dataset were collected from the UCI Machine Learning Repository, which is freely available for use. This dataset consists of 11055 Urls and 30 columns which were used to extract several website features [15].

Fig. 6, 7, 8 and 9 present the symbol count, ip count, url length and shortening url feature, respectively. Their characteristics is put on x axis and on axis respective numbers is kept. Here -1 means the feature has no impact to make it malicious, 0 means neutral and 1 it is used to identify the url as malicious. Here count for some features are given. we have tried this for all of the features which indicate how many individual classes for each of the features.

B. Impact Analysis

Fake url has a great impact on society.It has also a tremendous impact on almost every sectors of life like economy and politics. It has become a curse nowadays. So detecting the fake url is much needed thing to overcome this global issue. Some impacts are shown below:

1) Social and Environmental Impact: Phishing on social media sites like Instagram, LinkedIn, Facebook, or Twitter is referred to as social media phishing. Such an assault aims to take control of your social media account or to steal personal information. The negative impacts of phishing on a business are numerous and include financial loss, loss of intellectual property, reputational harm, and interruption of daily operations. These outcomes combine to reduce a company's value, sometimes with disastrous results.

2) Ethical Impact: The act of designing and carrying out simulations that are certain to cause stress and anxiety in all levels and roles of your employee base is ethical impact of phishing. The high rate of undesirable actions—such as clicking on a link, opening an attachment, or entering login information based on private or delicate topics—is not being generated on purpose or with predetermined objectives. Without the right context, these strategies are certain to endanger your program. An atmosphere of trust will be difficult to restore, regain, or establish.

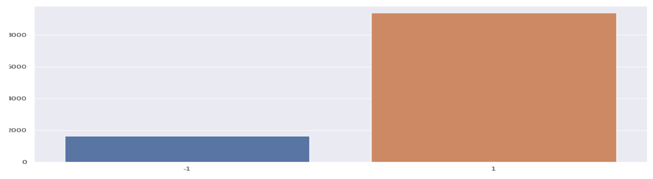


Fig. 6. Symbol Count.

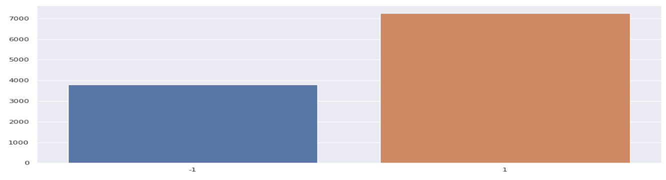


Fig. 7. Ip Count.

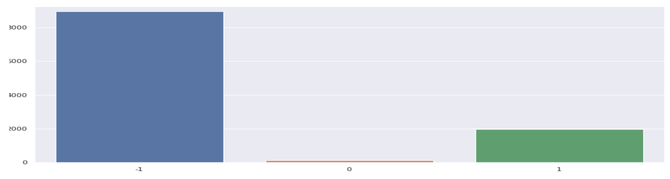


Fig. 8. Url Length.

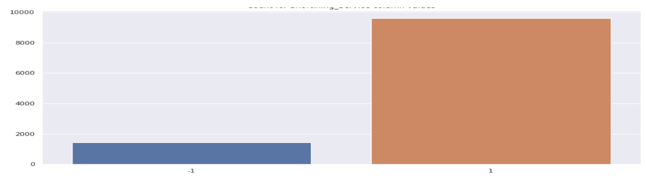
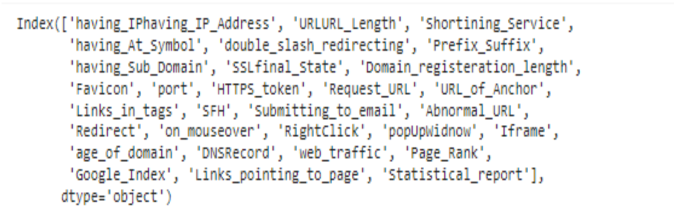


Fig. 9. Shortening Url.

C. Evaluation of Proposed Method

In this section, the method of our propose system will be shown step by step. Here we will discuss about our feature extraction technique, ensemble machine learning and finally the connection with the server.

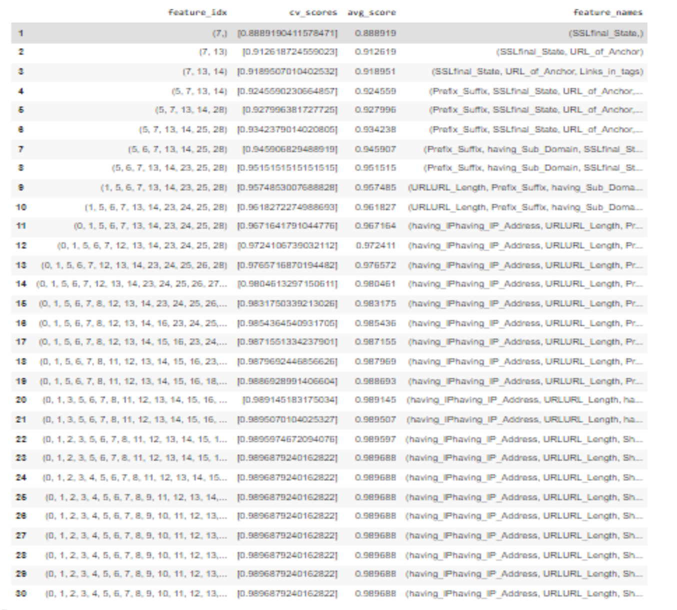
1) Feature Extraction: For feature extraction procedure we have used wrapper based, embedded method, Correlation and Coefficient, Information Gain, Chi square method. But checking the performance with ml techniques, we have come to a decision to use wrapper based feature extraction. It gives the highest result among all of them [7]. In this dataset, there were thirty columns before using feature extraction method. Fig. 10 shows the columns before extracting features.



```
Index(['having_IPhaving_IP_Address', 'URLURL_Length', 'Shortining_Service',
       'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix',
       'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length',
       'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor',
       'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL',
       'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe',
       'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank',
       'Google_Index', 'Links_pointing_to_page', 'Statistical_report'],
      dtype='object')
```

Fig. 10. Before Wrapper Used.

Then using wrapper, we got the performance of each of the subsets and came to know which subsets give the most accurate result which is shown in Fig. 11.

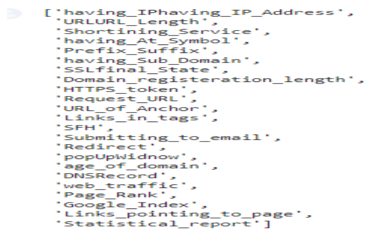


	feature_idx	cv_scores	avg_score	feature_names
1	(7,)	[0.8889190411578471]	0.888919	(SSLfinal_State,)
2	(7, 13)	[0.912618724559023]	0.912619	(SSLfinal_State, URL_of_Anchor)
3	(7, 13, 14)	[0.9189507010402532]	0.918951	(SSLfinal_State, URL_of_Anchor, Links_in_tags)
4	(5, 7, 13, 14)	[0.9245590230664857]	0.924559	(Prefix_Suffix, SSLfinal_State, URL_of_Anchor,...
5	(5, 7, 13, 14, 28)	[0.927996381727725]	0.927996	(Prefix_Suffix, SSLfinal_State, URL_of_Anchor,...
6	(5, 7, 13, 14, 25, 28)	[0.9342379014020805]	0.934238	(Prefix_Suffix, SSLfinal_State, URL_of_Anchor,...
7	(5, 6, 7, 13, 14, 25, 28)	[0.945906829488919]	0.945907	(Prefix_Suffix, having_Sub_Domain, SSLfinal_St...
8	(5, 6, 7, 13, 14, 23, 25, 28)	[0.9515151515151515]	0.951515	(Prefix_Suffix, having_Sub_Domain, SSLfinal_St...
9	(1, 5, 6, 7, 13, 14, 23, 25, 28)	[0.9574853007688828]	0.957485	(URLURL_Length, Prefix_Suffix, having_Sub_Doma...
10	(1, 5, 6, 7, 13, 14, 23, 24, 25, 28)	[0.9618272274988693]	0.961827	(URLURL_Length, Prefix_Suffix, having_Sub_Doma...
11	(0, 1, 5, 6, 7, 13, 14, 23, 24, 25, 28)	[0.9671641791044776]	0.967164	(having_IPhaving_IP_Address, URLURL_Length, Pr...
12	(0, 1, 5, 6, 7, 12, 13, 14, 23, 24, 25, 28)	[0.9724106739032112]	0.972411	(having_IPhaving_IP_Address, URLURL_Length, Pr...
13	(0, 1, 5, 6, 7, 12, 13, 14, 23, 24, 25, 26, 28)	[0.9765716870194482]	0.976572	(having_IPhaving_IP_Address, URLURL_Length, Pr...
14	(0, 1, 5, 6, 7, 12, 13, 14, 23, 24, 25, 26, 27...	[0.9804613297150611]	0.980461	(having_IPhaving_IP_Address, URLURL_Length, Pr...
15	(0, 1, 5, 6, 7, 8, 12, 13, 14, 23, 24, 25, 26,...	[0.9831750339213026]	0.983175	(having_IPhaving_IP_Address, URLURL_Length, Pr...
16	(0, 1, 5, 6, 7, 8, 12, 13, 14, 16, 23, 24, 25,...	[0.9854364540931705]	0.985436	(having_IPhaving_IP_Address, URLURL_Length, Pr...
17	(0, 1, 5, 6, 7, 8, 12, 13, 14, 15, 16, 23, 24,...	[0.9871551334237901]	0.987155	(having_IPhaving_IP_Address, URLURL_Length, Pr...
18	(0, 1, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 23,...	[0.9879692446856626]	0.987969	(having_IPhaving_IP_Address, URLURL_Length, Pr...
19	(0, 1, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 18,...	[0.9886928991406604]	0.988693	(having_IPhaving_IP_Address, URLURL_Length, Pr...
20	(0, 1, 3, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, ...	[0.989145183175034]	0.989145	(having_IPhaving_IP_Address, URLURL_Length, ha...
21	(0, 1, 3, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, ...	[0.9895070104025327]	0.989507	(having_IPhaving_IP_Address, URLURL_Length, ha...
22	(0, 1, 2, 3, 5, 6, 7, 8, 11, 12, 13, 14, 15, 1...	[0.9895974672094076]	0.989597	(having_IPhaving_IP_Address, URLURL_Length, Sh...
23	(0, 1, 2, 3, 5, 6, 7, 8, 11, 12, 13, 14, 15, 1...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
24	(0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
25	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14,...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
26	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
27	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
28	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
29	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...
30	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9896879240162822]	0.989688	(having_IPhaving_IP_Address, URLURL_Length, Sh...

Fig. 11. Subset by Wrapper.

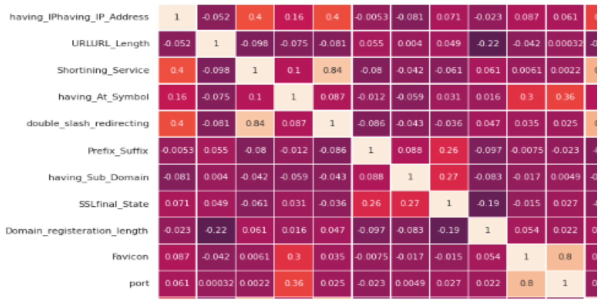
Fig. 12 shows the final outcomes which selected 23 features among all the features discrete rest of them.

2) Machine Learning Model: After using feature selection method, we have used this feature extraction method with several machine learning techniques as like Random forest, Light Gradient Boosted method and Category Boosting. We have come in a decision that Random Forest gives the highest accuracy with Coefficient Correlation. So in the next part, RF model will be discussed.



```
['having_IPhaving_IP_Address',
 'URLURL_Length',
 'Shortining_Service',
 'having_At_Symbol',
 'Prefix_Suffix',
 'having_Sub_Domain',
 'SSLfinal_State',
 'Domain_registeration_length',
 'HTTPS_token',
 'Request_URL',
 'URL_of_Anchor',
 'Links_in_tags',
 'SFH',
 'Submitting_to_email',
 'Redirect',
 'popUpWidnow',
 'age_of_domain',
 'DNSRecord',
 'web_traffic',
 'Page_Rank',
 'Google_Index',
 'Links_pointing_to_page',
 'Statistical_report']
```

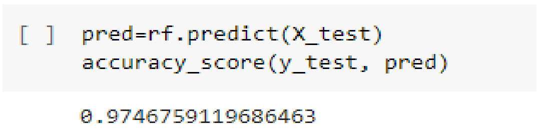
Fig. 12. Wrapper Final Selection.



having_IPhaving_IP_Address	1	-0.052	0.4	0.16	0.4	-0.0053	-0.081	0.071	-0.023	0.087	0.061
URLURL_Length	-0.052	1	-0.098	-0.075	-0.081	0.055	0.004	0.049	-0.22	-0.042	0.00032
Shortining_Service	0.4	-0.098	1	0.1	0.84	-0.08	-0.042	-0.061	0.061	0.0061	0.0022
having_At_Symbol	0.16	-0.075	0.1	1	0.087	-0.012	-0.059	0.031	0.016	0.3	0.36
double_slash_redirecting	0.4	-0.081	0.84	0.087	1	-0.086	-0.043	-0.036	0.047	0.035	0.025
Prefix_Suffix	-0.0053	0.055	-0.08	-0.012	-0.086	1	0.088	0.26	-0.097	-0.0075	-0.023
having_Sub_Domain	-0.081	0.004	-0.042	-0.059	-0.043	0.088	1	0.27	-0.083	-0.017	0.0049
SSLfinal_State	0.071	0.049	-0.061	0.031	-0.036	0.26	0.27	1	-0.19	-0.015	0.027
Domain_registeration_length	-0.023	-0.22	0.061	0.016	0.047	-0.097	-0.083	-0.19	1	0.054	0.022
Favicon	0.087	-0.042	0.0061	0.3	0.035	-0.0075	-0.017	-0.015	0.054	1	0.8
port	0.061	0.00032	0.0022	0.36	0.025	-0.023	0.0049	0.027	0.022	0.8	1

Fig. 13. Heatmap of RF with Coefficient.

Next in the Fig. 14 shows the highest accuracy done by random forest.



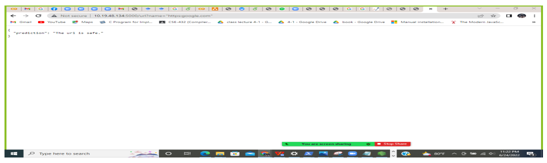
```
[ ]  pred=rf.predict(X_test)
     accuracy_score(y_test, pred)

     0.9746759119686463
```

Fig. 14. Accuracy.

Here in Fig. 13 shows the heatmap of random forest classification. Heatmap actually provides realtime analytics to understand the performance.

Here it shows that RF gives the highest result of 97.47% among all others and best suitable for proposed system.



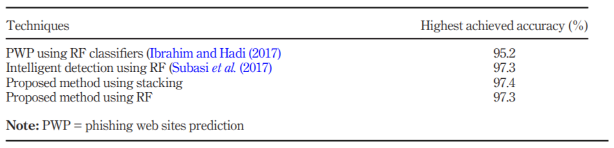
```
{
  "prediction": "The url is safe."
}
```

Fig. 15. Connecting with Server.

Finally in Fig. 15 shows our user interface connected with the server system.

D. Evaluation of Performance

In this section we will show a comparative analysis of the performance between our proposed methodologies and traditional approaches. At first we will show the analysis with different feature selection methods with different machine learning methods done by the base papers. Then we will show the performance of our proposed methods we have tried. Here, at first the performance of the traditional approaches are shown. In Fig. 16 shows the performance done by the previous paper [13]. The highest performance is done by using NN classifier + Random forest + bagging which is about 97.4.

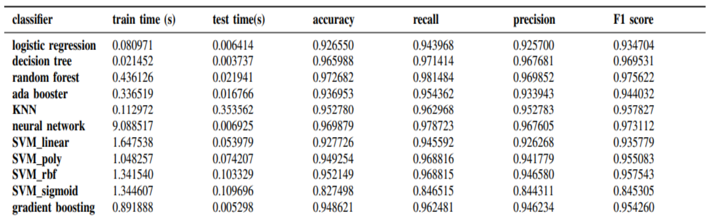


Techniques	Highest achieved accuracy (%)
PWP using RF classifiers (Ibrahim and Hadi (2017)	95.2
Intelligent detection using RF (Subasi *et al.* (2017)	97.3
Proposed method using stacking	97.4
Proposed method using RF	97.3

Note: PWP = phishing web sites prediction

Fig. 16. Performance by Paper [13].

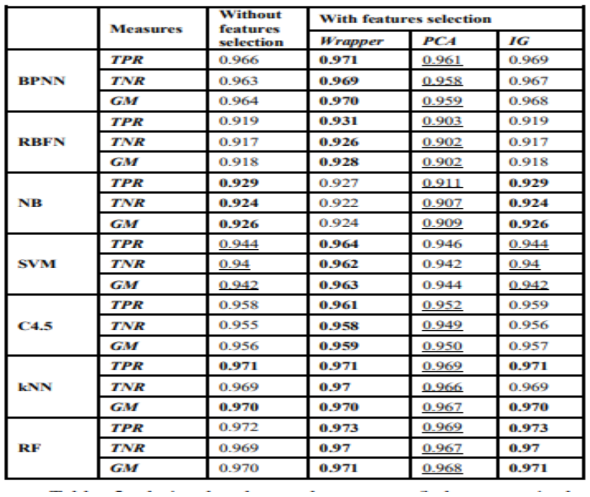
In Fig. 17 shows the performance done by the previous paper [14]. The highest performance is done by using Random forest which is about 97.3.



classifier	train time (s)	test time(s)	accuracy	recall	precision	F1 score
logistic regression	0.080971	0.006414	0.926550	0.943968	0.925700	0.934704
decision tree	0.021452	0.003737	0.965988	0.971414	0.967681	0.969531
random forest	0.436126	0.021941	0.972682	0.981484	0.969852	0.975622
ada booster	0.336519	0.016766	0.936953	0.954362	0.933943	0.944032
KNN	0.112972	0.353562	0.952780	0.962968	0.952783	0.957827
neural network	9.088517	0.006925	0.969879	0.978723	0.967605	0.973112
SVM_linear	1.647538	0.053979	0.927726	0.945592	0.926268	0.935779
SVM_poly	1.048257	0.074207	0.949254	0.968816	0.941779	0.955083
SVM_rbf	1.341540	0.103329	0.952149	0.968815	0.946580	0.957543
SVM_sigmoid	1.344607	0.109696	0.827498	0.846515	0.844311	0.845305
gradient boosting	0.891888	0.005298	0.948621	0.962481	0.946234	0.954260

Fig. 17. Performance by Paper [14].

In Fig. 18 shows the performance done by the previous paper [6]. The highest performance of the paper [6] is done by using random forest [19] with wrapper based feature selection which is about 97.3.



	Measures	**Without features selection**	**With features selection**		
			Wrapper	***PCA***	***IG***
BPNN	***TPR***	0.966	**0.971**	0.961	0.969
	TNR	0.963	**0.969**	0.958	0.967
	GM	0.964	**0.970**	0.959	0.968
RBFN	***TPR***	0.919	**0.931**	0.903	0.919
	TNR	0.917	**0.926**	0.902	0.917
	GM	0.918	**0.928**	0.902	0.918
NB	***TPR***	**0.929**	0.927	0.911	**0.929**
	TNR	**0.924**	0.922	0.907	**0.924**
	GM	**0.926**	0.924	0.909	**0.926**
SVM	***TPR***	0.944	**0.964**	0.946	0.944
	TNR	0.94	**0.962**	0.942	0.94
	GM	0.942	**0.963**	0.944	0.942
C4.5	***TPR***	0.958	**0.961**	0.952	0.959
	TNR	0.955	**0.958**	0.949	0.956
	GM	0.956	**0.959**	0.950	0.957
kNN	***TPR***	**0.971**	**0.971**	0.969	**0.971**
	TNR	0.969	**0.97**	0.966	0.969
	GM	**0.970**	**0.970**	0.967	**0.970**
RF	***TPR***	0.972	**0.973**	0.969	**0.973**
	TNR	0.969	**0.97**	0.967	**0.97**
	GM	0.970	**0.971**	0.968	**0.971**

Fig. 18. Performance of Paper [6].

The performance of our proposed system is given in Table I.

TABLE I. Performance of our Proposed System

Model	Features	Accuracy
Random Forest	Wrapper	97.377
Random Forest	No Feature Selection	97.40
Random Forest	Embedded	97.1058
Random Forest	Correlation Coefficient	97.47
Random Forest	Information Gain	96.65
Random Forest	Chi Square	97.467
Light Gradient Boosted	Wrapper	97.36
Light Gradient Boosted	No Feature Selection	97.40
Light Gradient Boosted	Embedded	97.07
Light Gradient Boosted	Correlation Coefficient	97.40
Light Gradient Boosted	Information Gain	96.77
Light Gradient Boosted	Chi Square	95.99
Category Boosting	Embedded	96.29
Category Boosting	Correlation Coefficient	97.13
Category Boosting	Information Gain	96.77
Category Boosting	Chi Square	95.9

In our proposed system, the performance are given in Fig. 19. Here the overall performance is better than the previous references. We can see that Random forest model with wrapper feature selection shows the accuracy of 97.377%, with no feature selection and also with correlation shows slightly better result as 97.40%, 97.47%. Light Gradient boosted and Catboost also have better performance with the features selection methods. But most accuracy are shown by using Random Forest with Coefficient based feature selection which is approximately 97.47% RF with Coefficient feature selection for our proposed system.

Then we have used react native to get the real life application using our proposed method. we have used flask server as backened. Then in the traditional approach we have extract a url which will be collected from user then extract it into our desired features. Next pickle file of ml is connected with that. Lastly, frontened is designed which can get data from user and give them a result using our machine learning approach.

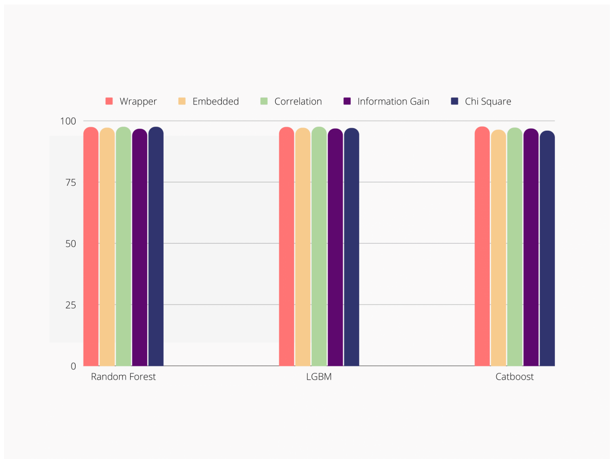
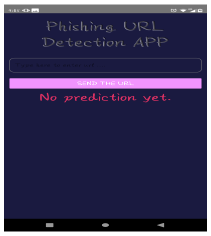


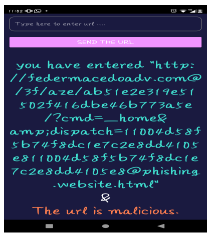
Fig. 19. Performance Measurement.

Fig. 20 shows the home page of the application. The input is given by the user and then it will be extracted and test whether its real url or fake one. The results of the input url can be fake or real which is given side by side. Fig. 21 showing that input url is malicious as well as fake one. Fig. 22 shows the input url is real one.



No prediction yet.

Fig. 20. Home Page.



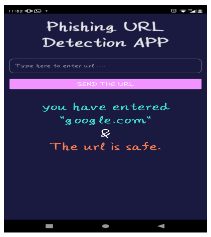
you have entered "http://federmacedoadv.com@/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&amp;dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html"
&
The url is malicious.

Fig. 21. Fake Url.

V. Conclusion and Future Work

We have precisely improved the performance using machine learning techniques with feature extraction methods comparing with the traditional approaches. we have found above 97.47% of performance using these methods. We have used both Machine Learning and feature extraction techniques to create the models. We have trained and validate from same distributed data to create the model and evaluated it from different distributed of test data. After the evaluation the best models are Random Forest combining with Coefficient feature extraction. Next we have used our machine learning model to create an application by using react native.The app can classify whether the url is fake or real by using our machine learning model.

Due to some hardware limitations and the limited amount of time, we couldn't use more existing methodologies like deep learning, neural network etc.We had to experiment continuously and carefully to build the models and use every possible methods. We couldn't apply all the features for our existing application. And the app is hosted on local server. So in future it's structure can be better by using all features We have already got from our machine learning model and also can improve the user interface. Different types of embedding techniques also can be used and the existing methods like deep learning, tuning, data cleaning can better its accuracy.



you have entered
"google.com"
&
The url is safe.

Fig. 22. Safe Url.

References

[1] S. Das, A. Kim, Z. Tingle, and C. Nippert-Eng, "All about phishing: Exploring user research through a systematic literature review," *arXiv preprint arXiv:1908.05897*, 2019.

[2] Z. A. Wen, Z. Lin, R. Chen, and E. Andersen, "What. hack: engaging anti-phishing training through a role-playing phishing simulation game," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.

[3] J.-N. Tioh, M. Mina, and D. W. Jacobson, "Cyber security training a survey of serious games in cyber security," in *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–5.

[4] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish," in *Proceedings of the 3rd symposium on Usable privacy and security*, 2007, pp. 88–99.

[5] G. Canova, M. Volkamer, C. Bergmann, and R. Borza, "Nophish: an anti-phishing education app," in *International workshop on security and trust management*. Springer, 2014, pp. 188–192.

[6] W. Ali, "Phishing website detection based on supervised machine learning with wrapper features selection," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 9, 2017.

[7] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[8] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[9] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, no. 3, pp. 203–228, 2000.

[10] L. Rokach, "Ensemble-based classifiers," *Artificial intelligence review*, vol. 33, no. 1, pp. 1–39, 2010.

[11] M. A. U. H. Tahir, S. Asghar, A. Zafar, and S. Gillani, "A hybrid model to detect phishing-sites using supervised learning algorithms," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2016, pp. 1126–1133.

[12] L. Wu, X. Du, and J. Wu, "Mobifish: A lightweight anti-phishing scheme for mobile phones," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2014, pp. 1–8.

[13] A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum, and M. Hamdani, "Phishing web site detection using diverse machine learning algorithms," *The Electronic Library*, vol. 38, no. 1, pp. 65–80, 2020.

[14] V. Shahrivari, M. M. Darabi, and M. Izadi, "Phishing detection using machine learning techniques," *arXiv preprint arXiv:2009.11116*, 2020.

[15] R. Mohammad, L. McCluskey, and F. Thabtah, "Uci machine learning repository: Phishing websites data set," *archive. ics. uci. edu*, 2015.

[16] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on j48 algorithm for data mining," *Proceedings of international journal of advanced research in computer science and software engineering*, vol. 3, no. 6, 2013.

[17] S. Srivastava and R. Kumar, "Design and implementation of disaster management application using react-native."

[18] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from urls," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.

[19] G. Biau, "Analysis of a random forests model," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 1063–1095, 2012.