PHISHING URL DETECTION USING ML

Sk.Khaja Mohiddin Basha[1], P.V.N.Sampath[2], G.Amarnadh[3],V.Siva Kumar Raja[4]
[1] Professor, [2, 3 & 4] Student
[1]sk.basha579@gmail.com, [2] sampathpasumarthi@gmail.com, [3] gunadalaamarnadh@gmail.com, [4] rajavengalasetty29659@gmail.com
Department of Computer Science and Engineering,
Narasaraopeta Engineering College, Narasaraopet, Andhra Pradesh, India

Abstract— **Phishing, a social engineering trick, increasing in today's internet age where websites can appears simple but is actually more complex. Attackers take advantage of these vulnerabilities to steal user data like logins and credit card numbers and several other confidential data. While we have a lot of countermeasures, phishing tactics constantly evolve to stay ahead. This is where Machine Learning emerges as a powerful weapon to fight against this phishing attack. Machine Learning techniques have shown great promise in tackling security issues. This paper focuses on Machine Learning's ability to recognize patterns in phishing attacks. By implementing various Machine Learning techniques to analyze URLs, the research leverages a dataset from UCI repository. The experiments, implemented in Python, aim to achieve an accuracy rate exceeding current methods in identifying phishing attempts. This would represent a significant jump in protecting users from these ever-evolving online threats.**

Keywords— Phishing url, Blacklist, Whitelist, detection, Machine learning, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM),AdaBoost, Neural Networks, Dataset, Python, Accuracy, AdaBoost, Neural Networks(NN), Dataset, Python, Anaconda environment, Scikit-learn library, Accuracy

I. INTRODUCTION

Phishing is a type of cybercrime where genuine entities are impersonated to get users personal and confidential data. Phishing attacks usually involve convincing people to submit their personal information on a fraudulent website or form via email, social media, or other communication channels. Phishing attacks can have serious outcomes, such as financial loss, reputational harm, and identity theft. These URLs can be challenging to identify since they frequently use URL-shortening services to hide the true destination or contain a very minor difference from the original URL. It is critical to identify these fraudulent URLs in order to stop attacks. URL detection techniques, such as manual inspection and blacklisting, are becoming more and more common.

These techniques have the potential to detect and stop attacks since they have an ability to automatically discover patterns that are hard to find manually. Machine learning models will effectively discriminate between phishing and legal URLs by examining a variety of factors, including context specific attributes, attribute based characteristics, and lexical driven data.

The detection of phishing URLs has been the subject of numerous machine learning approaches in the recent years; these range from conventional and small classification algorithms to more sophisticated deep learning techniques. In our research, we offer a ml-based methods for phishing URL detection that analyzes the contents once theURL is opened. The suggested model achieves great Jung Min Kang et al [6]. Lee et al released a technique, it uses a user's online activity to identify phishing. By using this strategy, users' profiles were kept up to date using a white list. The user's profile was automatically updated each time they visited a website. Here, an engine determined which website to use by calculating a score and comparing it to a cutoff point. The user profile entries and the specifics of the live website were used to compute the score.

accuracy in differentiating between phishing and genuine URLs by utilizing a number of features and a big dataset.

This project explores the potential of Machine Learning (ML) for building a strong and automated system for phishing URL detection. By analyzing features and contents extracted from URLs and training ML models on labeled datasets, we aim to develop a system that can accurately distinguish legitimate URLs from malicious ones. This will enhance online security and user confidence when navigating the web [1].

In line with the findings of the 3rd Microsoft Computing Safer Index Report, released in February 2014, the annual worldwide cost attributed to phishing could potentially escalate to $5 billion [2]. As user awareness diminishes, the efficacy of phishing attacks continues to rise.

The blacklist strategy is a comprehensive method for detecting phishing websites, entailing the inclusion of Internet Protocol (IP) addresses in the antivirus database and continuously updating the list of prohibited URLs. Cyber attackers employ various cunning techniques to deceive users into believing that a URL is legitimate by obfuscating its true nature. Heuristic detection, on the other hand, relies on recognizing patterns typically associated with phishing attacks, enabling the identification of zero-hour phishing attempts. Nonetheless, these distinctive features may not always be present in such attacks, resulting in a notable false positive rate in detection [3].

Security researchers are increasingly turning to machine learning methods to address the constraints of heuristic, whitelist, and blacklist approaches. Machine learning encompasses a variety of algorithms that leverage past data to predict or determine future data outcomes. Through this approach, algorithms scrutinize various characteristics of both blacklisted and legitimate URLs to accurately detect phishing websites, including those emerging on a zero-hour basis.

II. RELATED WORKS

A. The Whitelist and Blacklist Approaches

Minaxi Gupta et al [5]. presented a predictive blacklist technique. Utilizing heuristics and an effective matching technique, the system identifies novel phishing URLs. Heuristics amalgamate characteristics from blacklisted websites prone to phishing attacks to generate fresh URLs. A scoring mechanism is then employed by the matching algorithm to evaluate each URL. If the score surpasses a predefined threshold, the website is flagged as phishing. The scoring process involves comparing various components of the URL against those present in the blacklist, facilitating accurate determination.

B. Using Heuristics

Aaron Blum et al [7] conducted a study focusing on utilizing surface-level data extracted from URLs to develop a weighted learning algorithm with confidence for the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). The objective is to mitigate the risk associated with extracting host-based information by restricting the potential features to the character string of the URL. Each URL is represented as a binary feature vector. During testing, the online algorithm receives these vectors and assigns previously unseen URLs to their respective binary feature vectors. Subsequently, the algorithm produces a final output indicating whether the URL is potentially phishing or not.

M. H. Nguyen et al [8] introduced CANTINA+, a novel approach designed to identify phishing attempts. This method relies on machine learning techniques and focuses on utilizing the HTML Document Object Model (DOM), search engines, and third-party services. CANTINA+ incorporates eight innovative features aimed at enhancing its effectiveness. To minimize false positives and optimize runtime efficiency, two supplementary filters have been integrated. The first filter employs hashing to detect near- duplicate phishing attempts, while the second filter distinguishes authentic websites by identifying those without a login form.

Alsedrani, A et al [9] presented a study that utilized distinct classifiers for identifying and eliminating different phishing websites. They employed a combination of blacklisting techniques and host-based analysis. Feature values were extracted using host-based, popularity-based, and lexical-driven techniques, forming a comprehensive database. Various machine learning methods were then applied to analyze this database. After evaluating the classifiers, a particular one was selected and implemented using MATLAB.

]APWGM et al[10].noted the significance of the WHO is tool and the extent helpful it will be in expediting the global shutdown of fake sites in present era.

C. The Visual Similarity Method

URL and CSS matching-based hybrid method was provided byA. Mishra and B. B. Gupta et al [11]. This method can identify embedded noisy contents, such as an image on a webpage that maintains the webpage's visual similarity.]They compared the CSS similarity using the method that Jian Mao et al[12], Pei Li, Kun Li, Tao Wei, and Zhenkai Liang et al employed in and incorporated it into their own method Visual elements in web design can be categorized into two main groups: text content and text features. Font characteristics encompass aspects such as font family, font color, font size, and background color. By extracting page content from legitimate websites, attackers replicate the visual presentation of multiple sites through this approach. A browser-based plug-in called goldphish was proposed by Matthew Dunlop, Stephen Groat, and David Shelly et al in number-of-paragraph, number-of- script, length-of-title,has-h1, has-h2, has-h3, length-of- text, number-of-clickable-button, number-of-a, number-of- img,number-of-div, number-of-figure, has-footer, has-form, has-text-area, has-iframe, has-text-input, number-of-meta,

to detect phishing websites[13]. It recognizes the phony website using the emblems seen on websites. To trick internet users, the attacker can use the legitimate pic of the intended website. There are three steps to it:

• Logo harvesting: To extract the website logo from a dubious website, utilize Goldphish. After that, optical character recognition (OCR) software was used to turn into text.
• Authorized website harvesting: The harvested text undergoes a search engine query process. The search engine "Google" is commonly employed for this purpose as it reliably provides authentic websites as top results.
• Comparisons : Based on many attributes, the suspicious website is contrasted with the search engine's top result. Any domain that matches the live website is considered legitimate; if not, it is a phishing site.

II. PROPOSED SYSTEM

Our proposed system for the detection of phishing url mainly consists of the following steps those are shown in figure 1

(i) Data Collection
(ii) Data Preprocessing
(iii) Split Data
(iv) Defining ML Models
(v) Train the model
(vi) Evaluating model
(vii) Making Predictions
(viii) Calculating metrics
(ix) Output

In our experiment, we want to find the best algorithm for detecting phishing url. So, we tested different ones like Support Vector Machine, Random Forests, Decision Tree, AdaBoost, Gaussian Naïve Bayes, Neural Networks and K- Nearest Neighbors on a dataset that is taken from UCI repository. Then, we looked at the results to see which algorithm gave us the most accurate predictions. The proposed architecture is detailed in Figure 1.

Data Collection:

This is the first step in system. The data set consist of two categories those are structured and legitimate url and phishing url csv files which are collected from uci repository. In structured legitimate csv file it has 16062 instances and 45 attributes. In phishing there are 10525 instances and 45 attributes.

The attributes are has-title, has-input, has-button, has-image, has-submit', has-link, has-password, has-email-input, has-hidden-element, has-audio, has-video, number-of-inputs, number-of-buttons, number-of-images, number-of-option, number-of-list, number-of-th, number-of-tr, number-of-href, has-nav, 'has-object, 'has-picture, 'number-of-sources, number-of-span, number-of-table, URL, label.

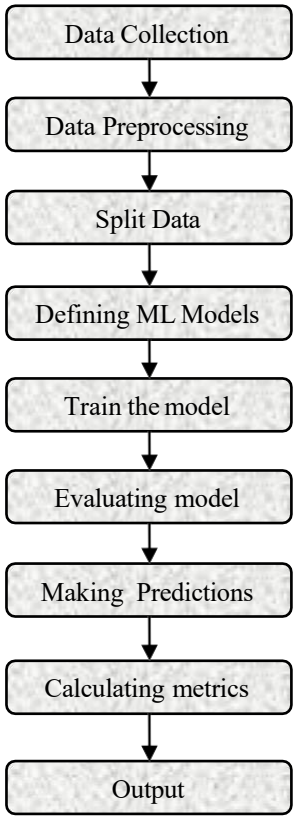

Fig.1 Flow of proposed system

Data Preprocessing

In our study data preprocessing is crucial for ensuring the dataset's quality and relevance in predicting phishing url.

- **Data Cleaning** We carefully deal with missing information and rid of any identical records to keep our dataset accurate and we find and fix any unusual data points to make sure they don't mess up our predictions.

Here in this study we dropped the url attribute because it does not help in model building then we checked for null values and then dropped unnamed column.

Here we plotted Heat map for misiing values before data cleaning and after data cleaning .

As shown in figure 2 we have lot of missing values in our dataset but after performing data cleaning operation the missing values are reduced to zero that is clearly visible in figure 3

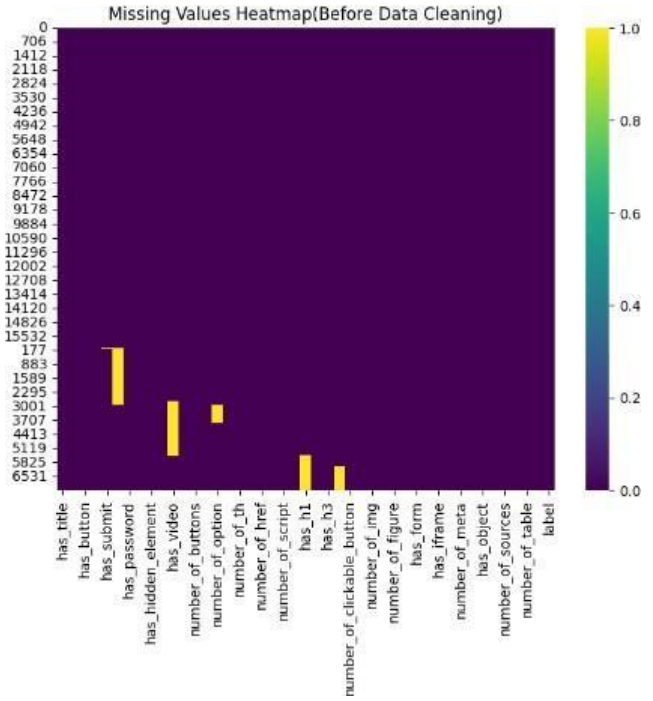

Fig.2 Missing values before data cleaning

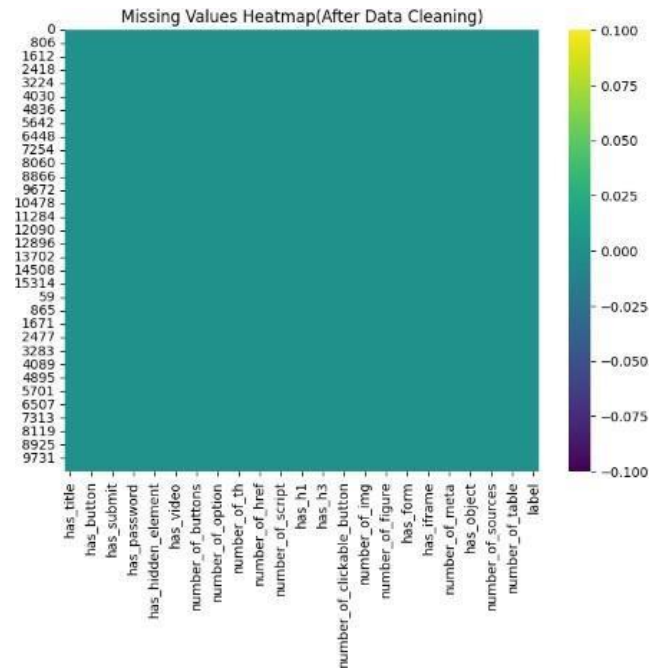

Fig.3 Missing values after data cleaning

- **Data CleaningDefining the Target Variable:** We define the target variable label that is used to indicate whether the url is legimate or phishing. This binary variable is helpful for our model s learn the pattern in finding the type of url.

.

Split Data

In this splitting the data the data is split into training data(x_train,y_train) and testing data(X_test,y_test)

a) X_train is features for training.
b) Y_train is corresponding tables for training.

c) X-test is features for testing
d) Y_test is actual labels for testing.

DEFINING ML MODELS

In this we defined seven supervised machine learning models for the highest accuracythose are Random Forest, Decision Tree, AdaBoost, Support Vector Machine, Gaussian Naive Bayes, Neural Network, K-Neighbours Classifier.

Training the Model

The next step that comes after defining ml models are training phase it is very crucial stage for a model performance. In previous steps we split the data into training and testing data. Using .fit() method we train the each and every model that we are using in this proposed system using training data.

Evaluating the Model

After training the model the next step is model evaluation. This step takes place with the help of testing data set that was defined in splitting the data step. The purpose of this step is to assess how well the model generalises to unseen data and helps us to understand it's strengths and weakness.

Making Predictions

Once the model is trained the next step is to make predictions on new, unseen data. Here the model processes input features and produces output that is either predictions or the classifications.

Calculating metrics

Calculating metrics is nothing but model assessment. Now it's time to evaluate its performance on untested data. This is crucial step entails evaluating its performance using a number of important metrics.

Accuracy:
Accuracy tells how often a model makes correct predictions. It's typically used for the classification tasks, where the model assigns data points to specific categories. Accuracy is calculated as the total number of correct predictions divided by the total number of predictions made.

Precision:
A measure of a model's positive prediction accuracy is called precision. It basically poses the question, "How many of the things the model classified as positive were in fact correct

Recall:
When evaluating a machine learning model, recall indicates how well your model locates all of the pertinent cases.

Misclassification:
The term "misclassification" in machine learning describes how frequently a classification model predicts something incorrectly. It is basically your model's error rate stated as a percentage of all predictions. A model that performs better is indicated by a reduced misclassification rate.

Sensitivity:
A statistic called sensitivity is used to evaluate how successfully a model detects positive cases. In essence, it indicates the percentage of real positive cases that the model accurately identified.

Specificity:
Particularly in classification problems, Specificity indicates how well your model detects true negatives. In essence, it calculates the percentage of negative cases that the model identified as negative with accuracy. A high specificity indicates that the model does not confuse negative cases for positive ones, which can be important in some circumstances.

F1_score:
A statistic called the F1-score is used in machine learning to assess how well categorization models perform. It considers recall as well as precision, which can occasionally clash. Recall measures how successfully the model recognized every single positive case, whereas precision measures how many of the positive predictions were true. By balancing these two and producing a single score between 0 and 1, the F1-score functions as a harmonic mean. When a model has a high F1-score, it is good at minimizing errors and finding positive cases.

False_positive_rate:
False_positive_rate is an indicator used to evaluate the performance of a classification model is the false positive rate (FPR). It indicates the percentage of negative cases that the model mistakenly interprets as positive.

False_negative_rate:
A statistic called false negative rate (FNR) is used to evaluate how well categorization models work. It indicates what percentage of real positive cases the model misclassified as negative.

Matthews_Correlation_Coefficient:
A statistic used in machine learning to evaluate the effectiveness of binary classification models is the Matthews Correlation Coefficient (MCC). It considers the proportions of each of the four categories—true positives, true negatives, false positives, and false negatives—within the data in a confusion matrix. In contrast to accuracy, which may be deceptive in datasets that are unbalanced, MCC takes into account both accurate and inaccurate classifications. A score of +1 for a perfect forecast, 0 for an arbitrary guess, and -1 for an entirely incorrect prediction are indicative of perfect predictions.

V. MACHINE LEARNING ALGORITHMS

In our study, we used seven machine learning to make predictions. The algorithms we used include:

K-neighbours classifier uses similarity to make predictions. A new data point is compared to its closest neighbors in the training set, and the average value or majority class of those neighbors is then assigned.

Decision tree is like a flowchart where each node represents a question or decision based on input features. It's a popular tool in machine learning used for both classification and regression tasks[14]. The tree splits the data into smaller groups based on features, aiming to create subsets that are as pure as possible, meaning they contain mostly one class or category.

GaussianNB is a classification algorithm that assumes your data follows a normal distribution. It predicts by analyzing the probability of each feature belonging to a specific class. While simple and fast, it requires continuous features and might not be ideal for complex data.

Support Vector Machine (SVM) is a machine learning algorithm used for classification and regression tasks. Its main objective is to find the best possible line or hyperplane that separates data points belonging to different classes with the widest margin[15]. SVM achieves this by identifying support vectors, which are data points closest to the decision boundary. These support vectors play a crucial role in determining the optimal separating
hyperplane.

Random forests are an ensemble learning method used for classification and regression tasks in machine learning[16]. They are constructed from a multitude of decision trees during training and output the mode of the classes (classification) or mean prediction (regression) of the individual tree.

AdaBoost short for Adaptive Boosting, is a machine learning technique that combines weak learners into a powerful ensemble. It trains weak learners one by one, focusing on data points that previous learners struggled with. This iterative approach transforms weak learners into a strong classifier, improving accuracy for various classification problems.

Neural networks which draw inspiration from the brain, employ interconnected nodes to learn from input. These "neurons" handle information similarly to a web, which enables them to take on challenging tasks like voice and image recognition.

VI. Experiment Environment

In our experiment environment for phishing url detection, we used google colab, a widely-used interactive computing environment for data analysis and machine learning tasks. We implemented seven different machine learning algorithms for finding phishing url's: Adaboost, Decision Tree Classifier (DTC), Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Neural networks, Gaussian naive bayes. To facilitate our experimentation, we leveraged several Python libraries, including: Scikit-learn (sklearn), Pandas, NumPy, Matplotlib and Seaborn etc. Additionally, to ensure robust evaluation of our models' performance and avoid overfitting, we employed k-fold cross-validation.

we created a robust, efficient experiment environment for phishing url detection. This setup enabled us to explore the effectiveness of different machine learning algorithms and make informed decisions regarding model selection and optimization strategies.
Moreover, In our study, we ensured that the software runs smoothly by specifying certain hardware and software requirements. For hardware, we recommend having at least an Intel Dual Core processor with a clock speed of 2.0GHz or higher, a minimum of 512GB of storage space on your hard disk, and 8GB of RAM for optimal performance. On the software side, you'll need a modern web browser such as Chrome, Windows 7 Server or a later version for your operating system, and Python installed to use COLAB. These specifications were chosen to guarantee that the software functions effectively and is compatible with most systems.

VII. Results

In our study, we conducted experiments to identify legitimate and phishing url's using various machine learning algorithms. We checked missing values and as well as we also drop the duplicates in the dataset that was taken from uci repository.

We applied seven different machine learning algorithms: Adaboost, Decision Tree Classifier (DTC), Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Neural networks, Gaussian naive bayes. Each algorithm was trained and tested on the preprocessed dataset to evaluate its predictive capabilities. Accuracy, precision, recall, sensitivity, specificity, misclassification, f1_score, false positive rate and false negative rate metrics were calculated for every model to quantitatively assess their performance.

The figure 4 below shows the comparison of Accuracy, precision, recall, sensitivity, specificity, misclassification, f1_score, false positive rate and false negative rate of each model. It clearly depicts that random forest has highest accuracy of and precision followed by decision tree ,adaboost and so on remaing models where GaussianNB achieved very low accuracy compared to other models

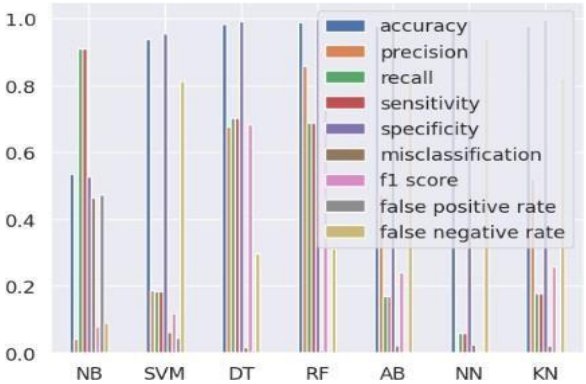

Fig. 4. Comparison of metrics

VIII. Discussion

Our experiments showed that the **Random Forest algorithm had the highest accuracy at 99%,** outperforming other models. Random Forest success is due to its ability to distinguish between phishing and legitimate url's by identifying complex decision boundaries. Although RF performed exceptionally well, other algorithms like Decision Tree Classifier (DTC), Adaboost, Neural Networks and K-Nearest Neighbors (KNN) also achieved good results, although their accuracy scores were slightly lower compared to Random Forest. We found that with the help of preprocessing techniques helped balance the data and improve the performance of all algorithms. Overall, our research highlights the effectiveness of machine learning in identifying the phishing url outcomes with the highest accuracy being 99%.

In our study, we took several steps to refine our model and improve its accuracy. Initially, we evaluated various machine learning algorithms using our original dataset.

Then we identified missing values in out dataset and replaces those values with the median values because after removing outliars it is better to replace with mean values but in out dataset there is no outliars so we replaced those missing values with median. Afer that we removed duplicate values to improve data quality.

i. Area Under Curve (AUC)

Table 1 shows AUC metric calculated for each model

Algorithm	AUC(%)
Random Forest	0.92
Decision Tree	0.61
AdaBoost	0.89
SVM	0.33
GaussianNB	0.82
Neural Networks	0.83
KNN	0.74

Table1. Area under Curve

The below figures form 5 to 11 shows the AUC and ROC curve for each model that we have used.

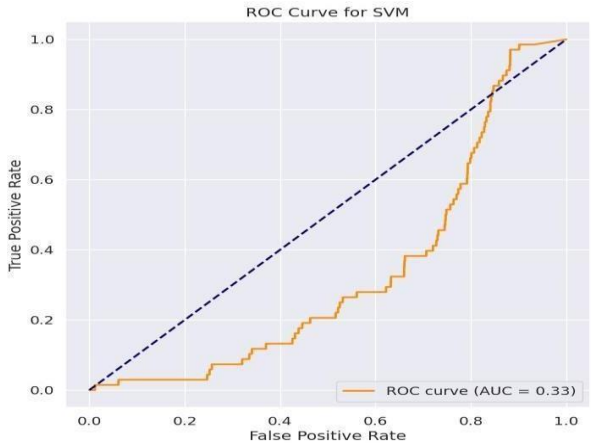

Fig-5:ROC and AUC curve for simple vector machine

The specific SVM model in the image has an AUC of 0.33, which signifies poor performance.

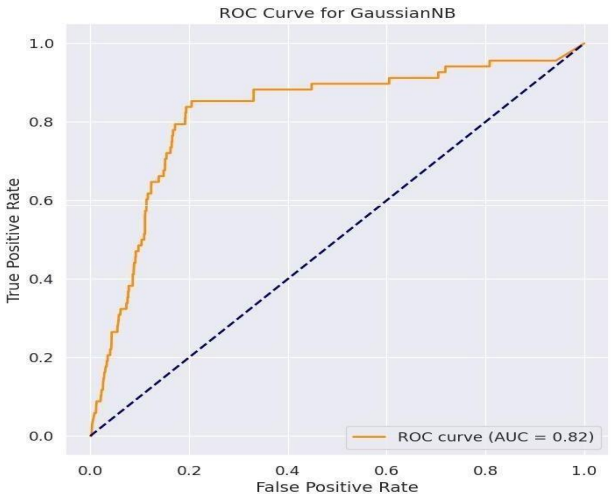

Fig-6: ROC and AUC curve for GaussianNB

The specific GNB model in the image has an AUC of 0.82, which signifies relatively good performance.

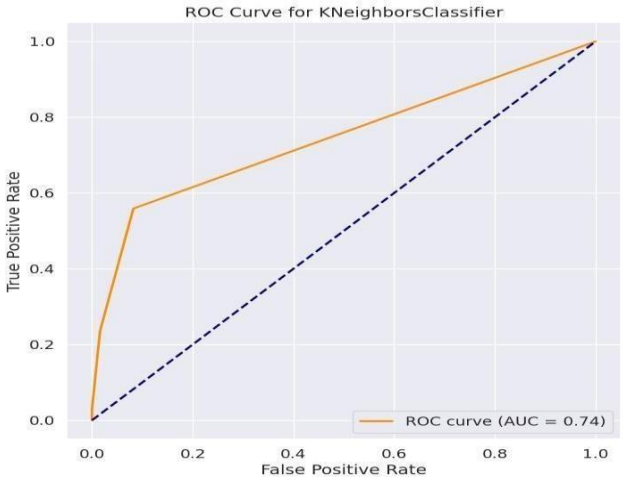

Fig-7: ROC and AUC curve for KneighborsClassifier

The specific KNN model in the image has an AUC of 0.74, which signifies fair performance.

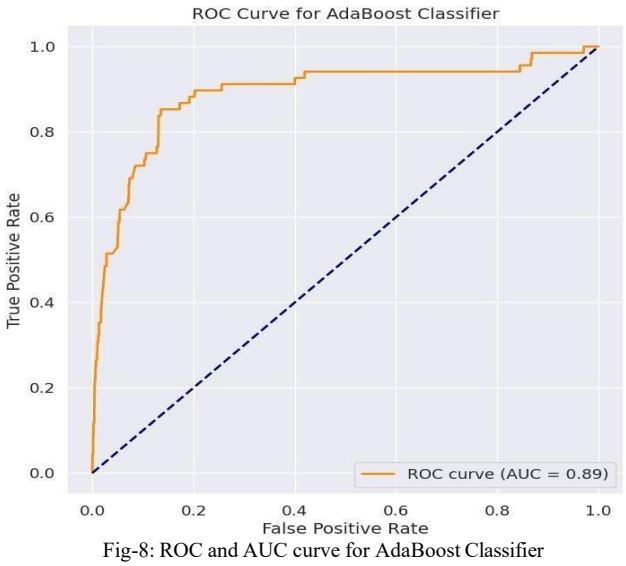

Fig-8: ROC and AUC curve for AdaBoost Classifier

The specific AdaBoost Classifier in the image has an AUC of 0.89, which signifies relatively good performance

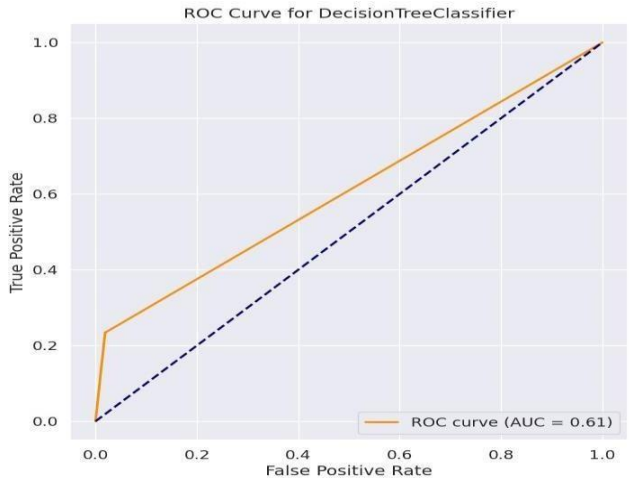

Fig-9: ROC and AUC curve for Decision Tree Classifier

The specific KNN model in the image has an AUC of 0.61, which signifies fair performance.

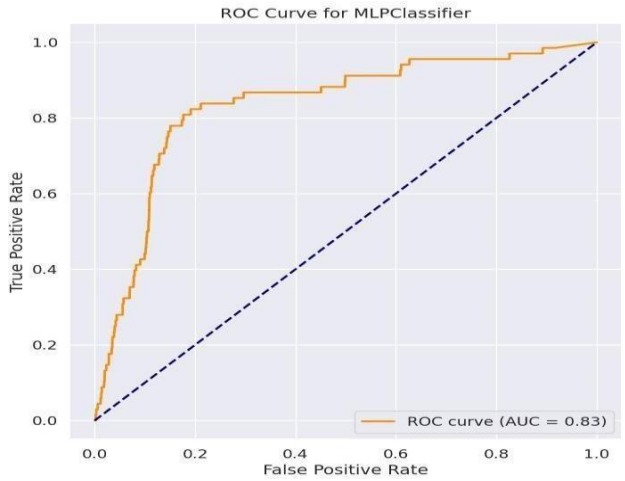

Fig-10: ROC and AUC curve for MLP Classifier

The specific MLP Classifier in the image has an AUC of 0.83, which signifies relatively good performance.

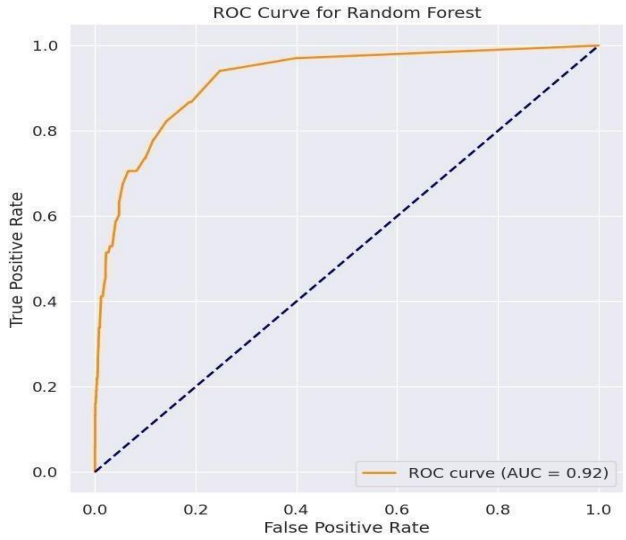

Fig-11: ROC and AUC curve for Random Forest

The AUC value for this specific curve is 0.92, which signifies good performance.

In Fig. 11 the ROC curves of each machine learning algorithm are illustrated, providing insights into their classification performance. The Area Under the Curve (AUC) serves as a pivotal metric, where higher values denote superior classifier performance. Random Forest achieves the highest AUC score of 0.92%, indicating its robust discriminative ability. Conversely, the Decision Tree classifier demonstrates the lowest SVM score of 0.33%, as depicted in Table 1.

These findings emphasize the superiority of Random Forest in detecting phishing url's, as evidenced by its impressive predictive Accuracy, precision, recall, sensitivity, specificity, misclassification, f1_score, false positive rate and false negative rate.

ii. Comparison of Accuracies

We've compared the testing accuracies of every model of our proposed system with the accuracies recorded in existing system in the Table2 below.

Approach	Proposed System Accuracy	Existing Model Accuracy
Random Forest	99.07	96.40
Decision Tree	98.37	94.75
AdaBoost	97.75	93.01
Support Vector Machine	93.84	-
Gaussian Naive Bayes	53.59	-
Neural Network	97.44	-
K-Neighbours Classifier	97.83	-

Table 2. Accuracy Comparison

From table 2 we can see the spike of testing accuracies from existing model to our proposed system. Random Forest has the highest testing accuracy of 99.07 and decision tree stays at the second position with an accuracy of 98.37. The improvement in accuracy is the result of data balancing techniques features in our proposed system.

iii. Classifiers Performance

In the Classifiers performance among all ml algorithms the Random Forest achieved the highest score when compared to all other models.RF achieved high in each and every metric that we calculated such as Accuracy, precision, recall, sensitivity, specificity, misclassification, f1_score, false positive rate and false negative rate.

The second place goes to Decision Tree it has an accuracy of 98% which is very good in solving or detecting real world scenarios .The next highest is K-neighbours classifiers achieved an accuracy of 97% which is quite impressive.

iv. K-fold Cross Validation:

We also applied K-fold cross validation in our study on every model with 5 folds and compared our testing accuracy with K fold accuracy in the below fig.12,We can see that the accuracies we get from k-fold cross-validation are bit higher than those from. This is because k-fold cross-validation involves training and testing the model multiple times on different parts of the data, providing a more thorough evaluation.

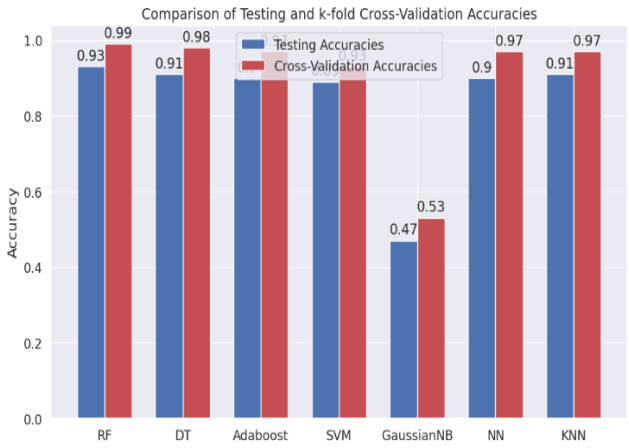

Fig.12. Comparison of k-fold and testing accuracy.

v. Confusion Matrix:

We created confusion matrices for each model using the testing data. These matrices help us see how well our models classify data by showing the number of correct and incorrect predictions. By analyzing these matrices, we can understand how accurate our models are in making predictions and identify areas for improvement. This process allows us to select the most effective model for our analysis and improve the overall reliability of our predictions. Here are the confusion matrices generated for every model.

i) Random Forest:

Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2908
1	0.68	0.19	0.29	102
accuracy			0.97	3010
macro avg	0.83	0.59	0.64	3010
weighted avg	0.96	0.97	0.96	3010

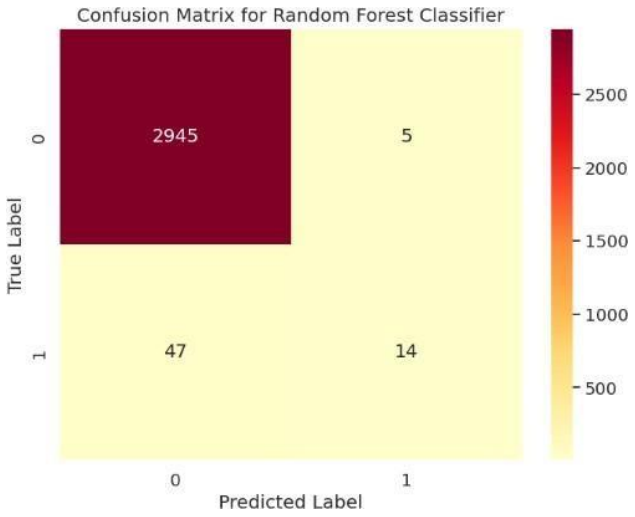

Fig. 13 Random Forest Confusion Matrix

Fig, 13 shows that the Random Forest model accurately identified 2945 positive cases and 14 negative cases. However, it incorrectly classified 5 negative cases as positive and missed 47 positive cases.

ii) Decision Tree

Decision Tree Classification Report:

	precision	recall	f1-score	support
0	0.97	0.98	0.98	2908
1	0.33	0.27	0.30	102
accuracy			0.96	3010
macro avg	0.65	0.63	0.64	3010
weighted avg	0.95	0.96	0.95	3010

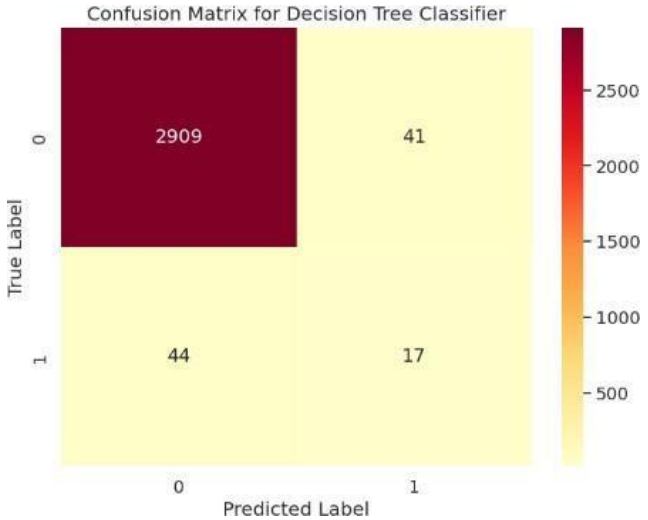

Fig. 14. Decision Tree Confusion Matrix

Fig. 14 shows that the Decision Tree model accurately identified 2909 positive cases and 17 negative cases. However, it incorrectly classified 41 negative cases as positive and missed 44 positive cases.

iii) AdaBoost Classifier:

ADA Boost Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2908
1	0.58	0.14	0.22	102
accuracy			0.97	3010
macro avg	0.78	0.57	0.60	3010
weighted avg	0.96	0.97	0.96	3010

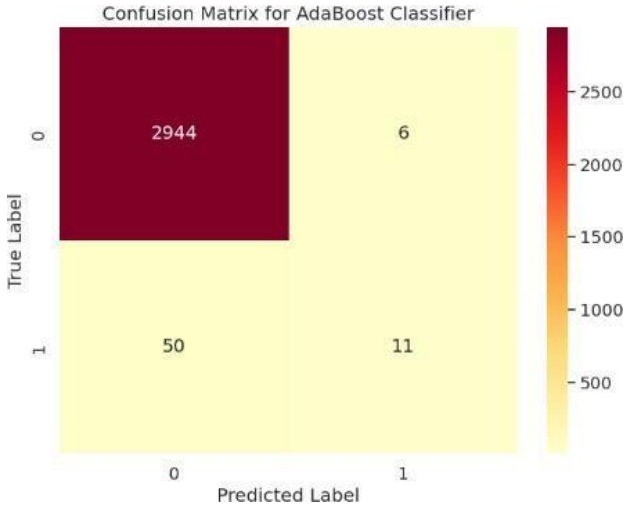

Fig. 15 AdaBoost Classifier Confusion Matrix

Fig.15 shows that the AdaBoost Classifier accurately identified 2944 positive cases and 11 negative cases. However, it incorrectly classified 6 negative cases as positive and missed 50 positive cases.

iv) SVM:

SVM Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2908
1	0.00	0.00	0.00	102
accuracy			0.96	3010
macro avg	0.48	0.50	0.49	3010
weighted avg	0.93	0.96	0.95	3010

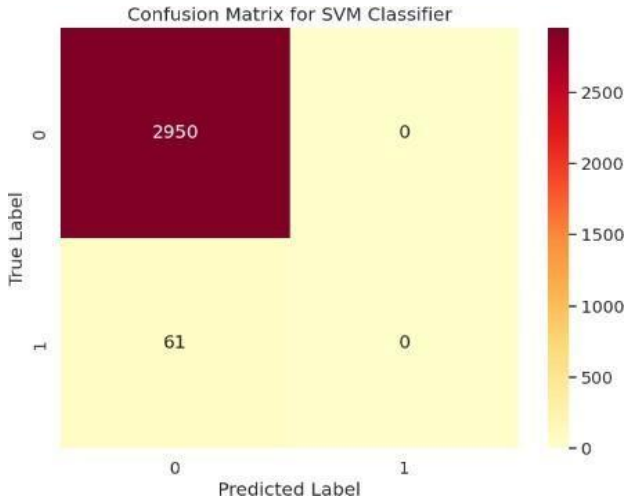

Fig. 16. SVM Confusion Matrix

Fig.16 shows that the SVM model accurately identified 2950 positive cases and 0 negative cases. However, it incorrectly classified 0 negative cases as positive and missed 61 positive cases.

v) GaussianNB:

Gaussian NB Classification Report:

	precision	recall	f1-score	support
0	0.99	0.51	0.67	2908
1	0.06	0.92	0.12	102
accuracy			0.52	3010
macro avg	0.53	0.72	0.40	3010
weighted avg	0.96	0.52	0.66	3010

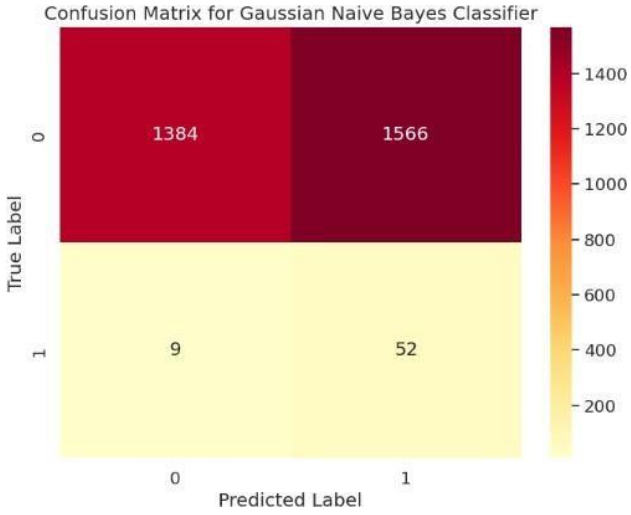

Fig. 17. Gaussian-NB Confusion Matrix

Fig.17 shows that the GaussianNB model accurately identified 1384 positive cases and 52 negative cases. However, it incorrectly classified 1566 negative cases as positive and missed 9 positive cases.

vi) Neural Networks:

Neural Network Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2908
1	0.52	0.11	0.18	102
accuracy			0.97	3010
macro avg	0.75	0.55	0.58	3010
weighted avg	0.95	0.97	0.96	3010

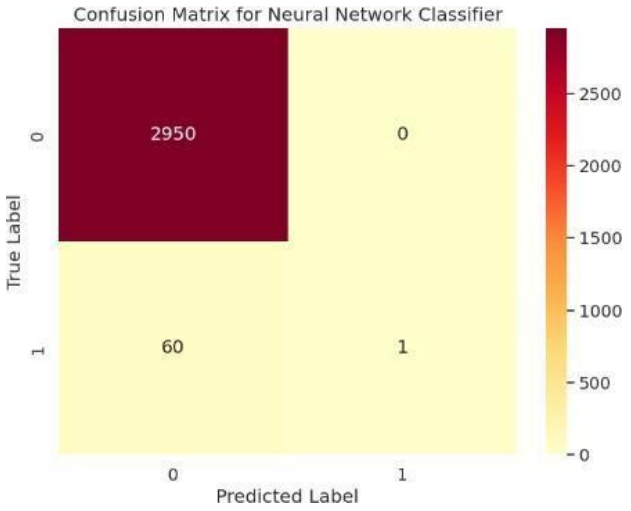

Fig. 18. Neural Networks Confusion Matrix

Fig.18 shows that the Neural Networks model accurately identified 2950 positive cases and 1 negative cases.

vii) KNN:

KNN Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2908
1	0.44	0.07	0.12	102
accuracy			0.97	3010
macro avg	0.70	0.53	0.55	3010
weighted avg	0.95	0.97	0.95	3010

However, it incorrectly classified 0 negative cases as positive and missed 60 positive cases.

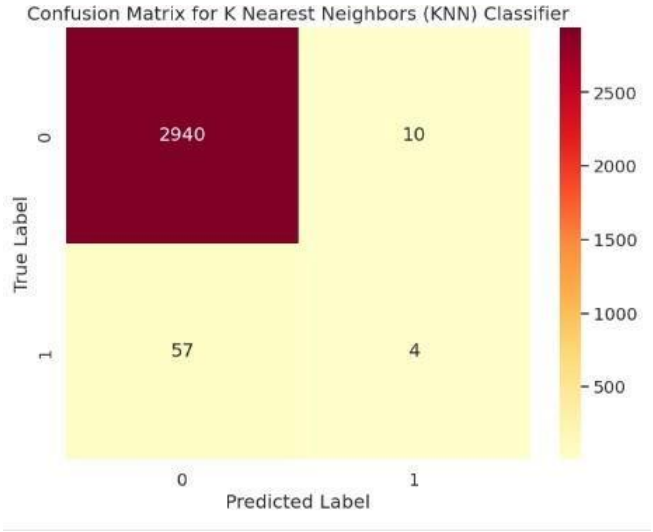

Fig. 19. KNN Confusion Matrix

Fig.19 shows that the KNN model accurately identified 2940 positive cases and 4 negative cases. However, it incorrectly classified 10 negative cases as positive and missed 57 positive cases.

IX. Conclusion

We examined the url dataset obtained from uci repository using seven different machine learning algorithms AdaBoost, Decision Tree Classifier (DTC), Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Neural networks, Gaussian naive bayes. Our analysis, conducted in Python with the scikit-learn library in the Google Colab environment, aimed to identify the most accurate and precise algorithm for phishing url detection. After thorough evaluation using metrics like Accuracy, precision, recall, sensitivity, specificity, misclassification, f1_score, false positive rate and false negative rate and Area Under the Curve (AUC), we determined that the Random Forest algorithm achieved the highest performance. With an accuracy of 99% RandomForest surpassed all other algorithms. These results highlight RF's effectiveness phishing url detection, showcasing its accuracy and precision. However, our study is limited to the UCI dataset, emphasizing the need for future research to validate these findings across various datasets

X. References

[1] Ollmann, G. (2004). "The phishing guide understanding & preventing phishing attacks". NGS Software Insight Security Research.

[2] https://resources.infosecinstitute.com/category/enterprise/phishing/the-phishing-landscape/phishing-data-attackstatistics/#gref

[3] Khonji, M., Iraqi, Y., & Jones, A. (2013). "Phishing detection: a literature survey". IEEE Communications Surveys & Tutorials, 15(4), 2091-2121.

[4] Mahajan, R., & Siddavatam, I. (2018). "Phishing website detection using machine learning algorithms". International Journal of Computer Applications, 181(23), 45-47.

[5] Zhang, Y., Hong, J. I., & Cranor, L. F. (2007, May). "Cantina: a content-based approach to detecting phishing web sites". In Proceedings of the 16th international conference on World Wide Web (pp. 639-648).

[6] Kang, J., & Lee, D. (2007, November). "Advanced white list approach for preventing access to phishing sites". In 2007 International Conference on Convergence Information Technology (ICCIT 2007) (pp. 491-496). IEEE.

[7] Blum, A., Wardman, B., Solorio, T., & Warner, G. (2010, October). "Lexical feature based phishing URL detection using online learning". In Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security (pp. 54-60).

[8] Nguyen, L. A. T., To, B. L., Nguyen, H. K., & Nguyen, M. H. (2013, October). "Detecting phishing web sites: A heuristic URL-based approach". In 2013 International Conference on Advanced Technologies for Communications (ATC 2013) (pp. 597-602). IEEE.

[9] Alswailem, A., Alabdullah, B., Alrumayh, N., & Alsedrani, A. (2019, May). "Detecting phishing websites using machine learning". In 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS) (pp. 1-6). IEEE.

[10] The Anti-Phishing Working Group, DNS Policy Committee;" Issues in Using DNS Whois Data for Phishing Site Take Down",The AntiPhishing Working Group Memorandum, 2011

[11] Mishra, A., & Gupta, B. B. (2014, August). "Hybrid solution to detect and filter zero-day phishing attacks". In Proceedings of the Second International Conference on Emerging Research in Computing, Information, Communication and Applications (pp. 373-379).

[12] Mao, J., Li, P., Li, K., Wei, T., & Liang, Z. (2013, September). "BaitAlarm: detecting phishing sites using similarity in fundamental visual features". In 2013 5th International Conference on Intelligent Networking and Collaborative Systems (pp. 790-795). IEEE.

[13] Dunlop, M., Groat, S., & Shelly, D. (2010, May). "Goldphish: Using images for content-based phishing analysis". In 2010 Fifth international conference on internet monitoring and protection (pp. 123-128). IEEE.

[14] http://dataaspirant.com/2017/01/30/how decision-treealgorithm-work

[15] https://www.kdnuggets.com/2016/07/support-vectormachines-simple-explanation

[16] lhttp://dataaspirant.com/2017/05/22/random-forestalgorithm-machine-learing