A Comparative Study of Malicious URL Detection: Regular Expression Analysis, Machine Learning, and VirusTotal API

Jason Misquitta[1][0009-0004-9906-3948] and Anusha K[1][0000-0002-8391-1744]

[1] School of Computer Science and Engineering, Vellore Institute of Technology, Vandalur-Kelambakkam Road, Chennai, 600127, Tamil Nadu, India.
lncs@springer.com

Abstract. In this paper, malicious URL detection was done in 3 ways. In the first method, self-written code was used that breaks up the URL into several fragments and then using functions and checkers analyses whether the URL is malicious. The second approach involved utilizing a fixed csv dataset containing website URLs from over 11,000 websites. Each entry in the dataset included 30 parameters describing the website and a class label indicating whether it was classified as a phishing website (1) or not (-1). 9 machine learning models were compared on this dataset to see which models gave the highest accuracy and F1-score. The last method was done using VirusTotal API Key. The user has to input the URL of a website and the code classifies it as malicious or not. A report is also generated if the URL is detected as malicious. The report is a compilation of the analysis of that website by several security vendors.

Keywords: malicious URL, antivirus, machine learning, VirusTotal API key, URL classification

1 Introduction

The Internet has become a crucial part of our lives, and it offers many benefits. However, with these benefits come threats, including viruses, malware, and other malicious activities. One of the most significant threats is through malicious URLs, which are links that appear legitimate but are designed to harm your computer or steal your personal information.

To counter this threat, antivirus software has become a crucial tool in safeguarding computers and networks against malicious URLs. Its primary purpose is to identify and eliminate malicious software from a computer or network. Additionally, antivirus software has the capability to detect malicious URLs and proactively prevent users from accessing them. Identifying malicious URLs is a pivotal aspect of cybersecurity, given the increasing sophistication and frequency of cyberattacks. Numerous methods and strategies have been suggested for detecting malicious URLs in recent times. This paper seeks to assess and contrast the efficiency of various approaches employed within this domain.

2 Literature Review

[1] presents an in-depth exploration of machine learning's application in identifying malicious URLs. The survey encompasses diverse methodologies, algorithms, and techniques employed in this domain. It investigates the challenges and advancements in URL-based threat detection, serving as a valuable resource for researchers and practitioners working on cybersecurity and threat mitigation. The authors explore the changing nature of cybersecurity risks and the contribution of machine learning in enhancing online security.

[2] focuses on employing machine learning to identify malicious URLs through lexical features. The study explores the utilization of various linguistic and contextual attributes present in URLs for effective detection. The authors propose a methodology

that involves feature extraction and classification using machine learning algorithms. The research aims to enhance cybersecurity by offering a technique to swiftly recognize potentially harmful URLs. The findings contribute to the ongoing efforts in devising robust tools for safeguarding online environments against evolving cyber threats.

[3] presents a comprehensive exploration of malicious URL detection strategies, employing a synergistic blend of machine learning and heuristic methodologies. The research investigates the effectiveness of these combined techniques in accurately identifying potentially harmful URLs. Leveraging machine learning algorithms for intricate pattern recognition and incorporating rule-based heuristics for deeper analysis, the proposed approach aims to achieve enhanced precision in malicious URL identification. The study significantly contributes to the ongoing discourse on cybersecurity. By addressing the evolving challenges posed by cyber threats, this research offers valuable insights and practical implications for safeguarding digital environments.

[4] addresses the opacity surrounding the labeling process of online scan engines like VirusTotal, which researchers heavily rely on to categorize malicious URLs and files. The paper aims to unravel the intricacies of the labeling generation process and assess the reliability of scanning outcomes. Focusing on VirusTotal and its 68 third-party vendors, the study centers on the labeling process for phishing URLs. Through a comprehensive investigation involving the establishment of mimic PayPal and IRS phishing websites and subsequent URL submissions for scanning, the authors analyze the network traffic and dynamic label changes within VirusTotal. The study uncovers significant insights into VirusTotal's operational mechanisms and label accuracy. Findings reveal challenges faced by vendors in identifying all phishing sites, with even the most proficient vendors failing to detect 30% of phishing sites. Moreover, inconsistencies emerge between VirusTotal scans and some vendors' in-house scanners, shedding light on the need for more robust methodologies to evaluate and effectively utilize VirusTotal's labels.

Phishing represents a type of online social manipulation aimed at stealing digital identities by posing as legitimate entities. This entails sending malicious content through channels like emails, chats, or blogs, often embedded with URLs leading to deceptive websites designed to extract private information from victims. The main objective of [5] is to construct a system capable of scrutinizing and categorizing URLs, primarily to identify phishing attempts. Rather than resorting to traditional methods involving website visits and subsequent feature extraction, the proposed strategy focuses on URL analysis. This not only ensures a level of distance between the attacker and the target but also demonstrates faster performance when compared to alternative methods like conducting internet searches, fetching content from destination websites, and utilizing network-level characteristics as observed in prior research. The research extensively explores various facets of URL analysis. This encompasses performance assessments conducted on balanced and imbalanced datasets, covering both controlled and real-time experimental scenarios. Additionally, the study explores the distinctions between online and batch learning approaches.

[6] examines a lightweight strategy for detecting and categorizing malicious URLs based on their specific attack types. The efficacy and efficiency of using lexical analysis as a proactive means of identifying these URLs is demonstrated. The study identifies a comprehensive set of essential features required for precise categorization. The accuracy of this approach is evaluated using a dataset comprising over 110,000 URLs. An examination is conducted to assess the effects of obfuscation methods.

Due to mobile screens' limited size, manually verifying phishing URLs sent via text messages becomes challenging. Clicking a smishing URL can either direct users

to phishing sites or attempt to implant harmful software. [7] aims to counter Smishing attacks on Android devices through a novel application integrating established phishing APIs. This background-running app validates URL authenticity in received text messages. Five APIs were tested on a 1500 URL dataset, revealing the VirusTotal API's 99.27% accuracy but slower response (12-15 seconds) and the Safe-Browsing API's 87% accuracy with swift (0.15ms) response. Depending on the application's urgency or security emphasis, the choice of API varies.

[8] introduces "Obfuscapk," an open-source tool designed for enhancing the security of Android apps through obfuscation. Obfuscation is crucial to protect apps from reverse engineering. The tool operates as a "black-box," obfuscating apps without requiring their original source code. It applies various techniques, including renaming classes, methods, variables, and encrypting strings, making the code harder to decipher. As a response to the vulnerability of APK files to reverse engineering, Obfuscapk contributes significantly to mobile app security. The paper outlines the tool's functionalities, emphasizes its role in safeguarding intellectual property, and showcases how it mitigates potential security vulnerabilities. Obfuscapk's approach serves as a valuable contribution to the field of Android app security.

Due to a lack of security awareness, numerous web applications are vulnerable to web attacks. Addressing this, there's a pressing need to bolster web application reliability by effectively identifying malicious URLs. While past efforts have predominantly employed keyword matching for this purpose, such an approach lacks adaptability. [9] introduces a groundbreaking technique that merges statistical analysis via gradient learning with the extraction of features using a sigmoidal threshold. The study utilizes naïve Bayes, decision tree, and SVM classifiers to evaluate the effectiveness and efficiency of this method. The empirical outcomes emphasize its strong ability to detect patterns, yielding an accuracy rate that exceeds 98.7%. Notably, the method has been deployed online for large-scale detection, successfully analyzing around 2 TB of data daily.

The surge in internet-connected devices like smartphones, IoT, and cloud networks has led to increased phishing attacks exploiting human vulnerabilities. Unlike system-focused attacks, phishing deceives users into revealing personal data. Addressing this, a streamlined phishing detection approach utilizing only nine lexical features has been introduced in [10]. Unlike resource-intensive methods, this approach is suitable for constrained devices. It applied Random Forest and was tested on the ISCXURL-2016 dataset with 11,964 instances.

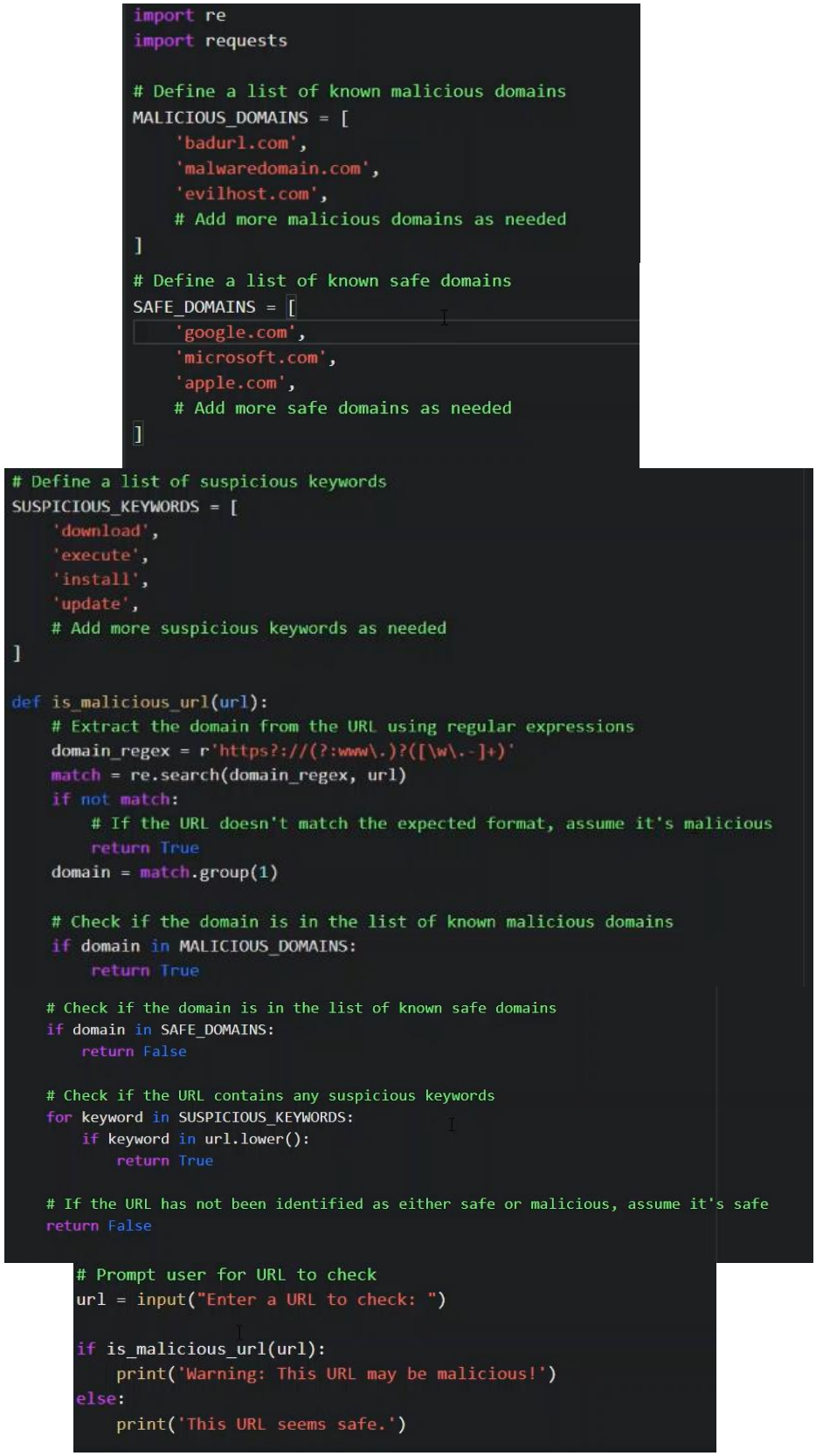
3 Methodology

In this paper, malicious URL detection was done in 3 separate unique ways. Each method has been described below.

3.1 Segmenting and Analyzing URL

In the first method, the following self-written code in Fig 1. was used for breaking up the URL into different fragments and then evaluating it using various functions. The code checks if the URL belongs to a Malicious or Safe Domain. It checks for suspicious keywords. [11] Parsing is done through regular expression to see if the URL matches the correct format. The path and file name within the URL was checked for additional clues about the content. For instance, URLs with file extensions like .exe, .zip, or .js might indicate potential malware downloads. [12]

4



```
import re
import requests

# Define a list of known malicious domains
MALICIOUS_DOMAINS = [
    'badurl.com',
    'malwaredomain.com',
    'evilhost.com',
    # Add more malicious domains as needed
]

# Define a list of known safe domains
SAFE_DOMAINS = [
    'google.com',
    'microsoft.com',
    'apple.com',
    # Add more safe domains as needed
]

# Define a list of suspicious keywords
SUSPICIOUS_KEYWORDS = [
    'download',
    'execute',
    'install',
    'update',
    # Add more suspicious keywords as needed
]

def is_malicious_url(url):
    # Extract the domain from the URL using regular expressions
    domain_regex = r'https?://(?:www\.)?([\w\.-]+)'
    match = re.search(domain_regex, url)
    if not match:
        # If the URL doesn't match the expected format, assume it's malicious
        return True
    domain = match.group(1)

    # Check if the domain is in the list of known malicious domains
    if domain in MALICIOUS_DOMAINS:
        return True

    # Check if the domain is in the list of known safe domains
    if domain in SAFE_DOMAINS:
        return False

    # Check if the URL contains any suspicious keywords
    for keyword in SUSPICIOUS_KEYWORDS:
        if keyword in url.lower():
            return True

    # If the URL has not been identified as either safe or malicious, assume it's safe
    return False

# Prompt user for URL to check
url = input("Enter a URL to check: ")

if is_malicious_url(url):
    print('Warning: This URL may be malicious!')
else:
    print('This URL seems safe.')
```

Fig. 1. Code for analyzing URL

3.2 Comparing 9 Machine Learning Models

In the second method, a dataset from Kaggle was used : https://www.kaggle.com/eswarchandt/phishing-website-detector. A small part of the dataset is shown in Fig.2. It has 11054 samples with 32 features.

R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
InfoEmail	Abnormall	WebsiteFo	StatusBar(	DisableRig	UsingPopu	IframeRed	AgeofDom	DNSRecor	WebsiteTr	PageRank	GoogleInd	LinksPoint	StatsRepo	class
1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1
-1	-1	0	1	1	1	1	1	-1	1	-1	1	0	-1	-1
1	1	0	1	1	1	1	-1	-1	1	-1	1	-1	1	-1
1	1	0	-1	1	-1	1	-1	-1	0	-1	1	1	1	1
-1	-1	0	1	1	1	1	1	1	1	-1	1	-1	-1	1
-1	-1	0	1	1	1	1	1	-1	-1	-1	1	0	-1	-1
1	1	0	1	1	1	1	-1	-1	0	-1	1	0	1	-1
1	1	0	1	1	1	1	1	-1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	-1	0	-1	1	0	1	-1
-1	-1	0	1	1	1	1	-1	1	1	1	1	-1	-1	1
-1	-1	0	1	1	1	1	-1	-1	-1	-1	1	0	-1	-1
1	1	0	-1	1	-1	1	1	-1	-1	-1	1	0	1	-1
1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1
1	1	0	1	1	1	1	1	-1	1	-1	1	-1	1	1
1	1	0	1	1	1	1	1	-1	-1	-1	1	0	1	-1
-1	-1	0	1	1	1	1	1	-1	0	-1	1	1	-1	-1
-1	-1	0	1	1	1	1	-1	1	1	-1	1	1	-1	-1
-1	-1	0	1	1	1	1	1	-1	-1	1	1	-1	-1	1
-1	-1	0	-1	-1	-1	-1	1	-1	0	-1	1	0	-1	1
-1	-1	0	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1
-1	-1	0	1	1	1	1	-1	1	-1	-1	1	0	-1	1
1	1	0	1	1	1	1	1	1	0	-1	1	-1	1	1

Fig. 2. A small section of the Dataset

The final attribute serves as the class label, indicating whether it categorizes the website as a phishing site (1) or not (-1). The following models were trained on this dataset and then compared on evaluation metrics.

- **Gradient Boosting Classifier:** Gradient Boosting Classifier serves as an ensemble learning method that merges multiple weak models (decision trees) to establish a more robust classifier. It follows a sequential training process, wherein each tree aims to rectify the mistakes made by the preceding trees. Renowned for its exceptional accuracy and proficiency in handling extensive datasets, this algorithm finds valuable application in malicious URL detection and other domains where precision and scalability are essential.

- **CatBoost Classifier:** CatBoost Classifier is a gradient boosting algorithm that is optimized for categorical data. It handles categorical variables efficiently by encoding them as numerical variables and reducing the impact of noisy variables. It also performs well on datasets with missing values, without requiring imputation. [13]

- **Random Forest:** Random Forest represents an ensemble learning algorithm utilized for malicious URL detection, which constructs numerous decision trees and amalgamates their predictions. To mitigate overfitting, each tree is created using a random subset of the features. Known for its straightforwardness, impressive accuracy, and capacity to manage extensive datasets, Random Forest serves as a valuable tool in the field of security to effectively identify potentially harmful URLs. [14]

- **Multi-layer Perceptron:** Multi-layer Perceptron is a neural network structure comprising multiple layers of perceptrons. It is capable of learning complex non-

linear relationships between inputs and outputs. It is a versatile algorithm that can be used for classification, regression, and other machine learning tasks. [15]

- **Support Vector Machine:** Support Vector Machine (SVM) represents a binary classification technique that seeks to identify the best hyperplane to separate two classes while maximizing the margin between them. SVM is particularly renowned for its proficiency in handling datasets with numerous dimensions and those that are not linearly separable.

- **Decision Tree:** Decision Tree is a classification algorithm that builds a model resembling a tree to make decisions and determine their potential outcomes. This straightforward and easy-to-understand algorithm is capable of handling both categorical and continuous data. Its versatility makes it a valuable tool in various domains. [16]

- **K-Nearest Neighbors:** K-Nearest Neighbors (KNN) is a non-parametric classification technique that assigns a class label to an instance based on the class labels of its k-closest neighbors within the training dataset. This straightforward algorithm does not involve any training phase and is capable of addressing multi-class problems. Nonetheless, its performance may be hindered by its relatively slow and memory-intensive nature, particularly when applied to large datasets. [17]

- **Logistic Regression:** For the purpose of malicious URL detection, Logistic Regression can be employed as a linear classification algorithm. By utilizing a logistic function and input features, it can calculate the probability of a URL being malicious or benign. Due to its simplicity and interpretability, Logistic Regression can be a valuable tool for identifying potentially harmful URLs in various security applications. [18]

- **Naive Bayes Classifier:** The Naive Bayes Classifier is a classification method that relies on probabilities and makes an assumption that the input features are conditionally independent when given the class. It employs Bayes' theorem to compute the posterior probability for each class, taking into account the provided features. Naive Bayes is renowned for its simplicity and efficiency, making it suitable for handling high-dimensional data. This algorithm finds application in various fields like spam filtering, sentiment analysis, and document classification.

The main aim of the second method is to see which machine learning models are best suited to malicious URL classification.

3.3 Using VirusTotal API key

The third method involved the use of VirusTotal API key to check whether the URL is malicious or not.

VirusTotal is a no-cost web-based platform that examines files and web addresses for viruses, worms, trojans, and other types of malicious software. Users can submit a file or URL to VirusTotal, and the service will scan it using more than 70 different antivirus engines and other malware detection tools. VirusTotal also offers an API that allows developers to integrate its malware scanning capabilities into their own applications. [19] By using a VirusTotal API key, developers can submit files or

URLs programmatically and retrieve the results for further analysis or processing. The VirusTotal API key can also be used to retrieve additional information about malware samples, such as their behavior and network activity. VirusTotal's API is widely used in the cybersecurity industry for threat intelligence, malware analysis, and incident response. [20]

4 Results

4.1 Results of Self-Written Code

As seen in Fig.3, the first method gave the following outputs for various URLs.

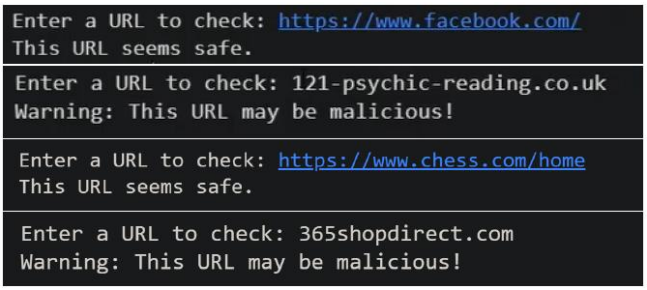


Fig. 3. Output of Code

The outputs were highly accurate (close to 90%) in classifying the URLS. Thus the self-written code proved to be quite effective in detecting malicious URLs.

4.2 Results of Comparison between 9 Machine Learning Models

Table 1. Evaluation Metrics Comparison of 9 Models

Models	Accuracy	F1 Score	Recall	Precision
Gradient Boosting Classifier	0.974	0.977	0.994	0.986
CatBoost Classifier	0.972	0.975	0.994	0.989
Random Forest	0.966	0.969	0.993	0.989
Multi-layer Perceptron	0.966	0.969	0.977	0.993
Support Vector Machine	0.964	0.968	0.980	0.965
Decision Tree	0.960	0.964	0.991	0.993
K-Nearest Neighbors	0.956	0.961	0.991	0.989
Logistic Regression	0.934	0.941	0.943	0.927
Naive Bayes Classifier	0.605	0.454	0.292	0.997

Table 1 was obtained after running all the machine learning models mentioned in the second method. The table displays the evaluation metrics (accuracy, F1-score, recall, precision) of all 9 models. It is evident that the accuracy, F1-score and recall are extremely high for all the models except for Naive Bayes Classifier. This indicates that all these models (except Naive Bayes) are equally useful and suited to malicious URL detection.

Training and test accuracy was plotted for each model. In Fig.4, the accuracy curves for the best model (Gradient Boosting Classifier) are shown.

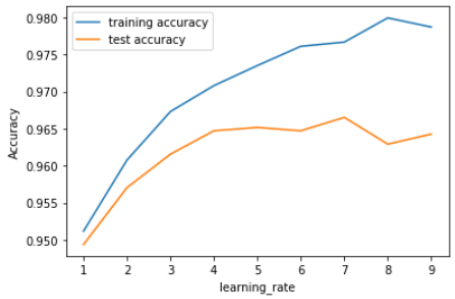


Fig. 4. Plotting Accuracy of Gradient Boosting Classifier

To understand the pairwise relationships between different features in the dataset, we applied the Seaborn Pairplot as seen in Fig.5.

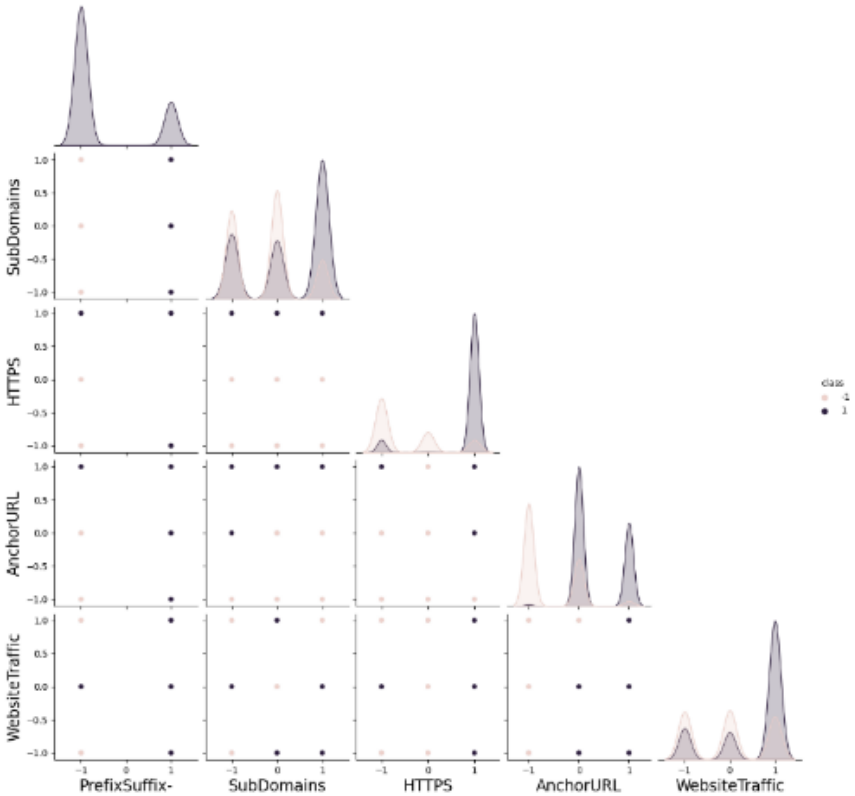


Fig. 5. Pairplot

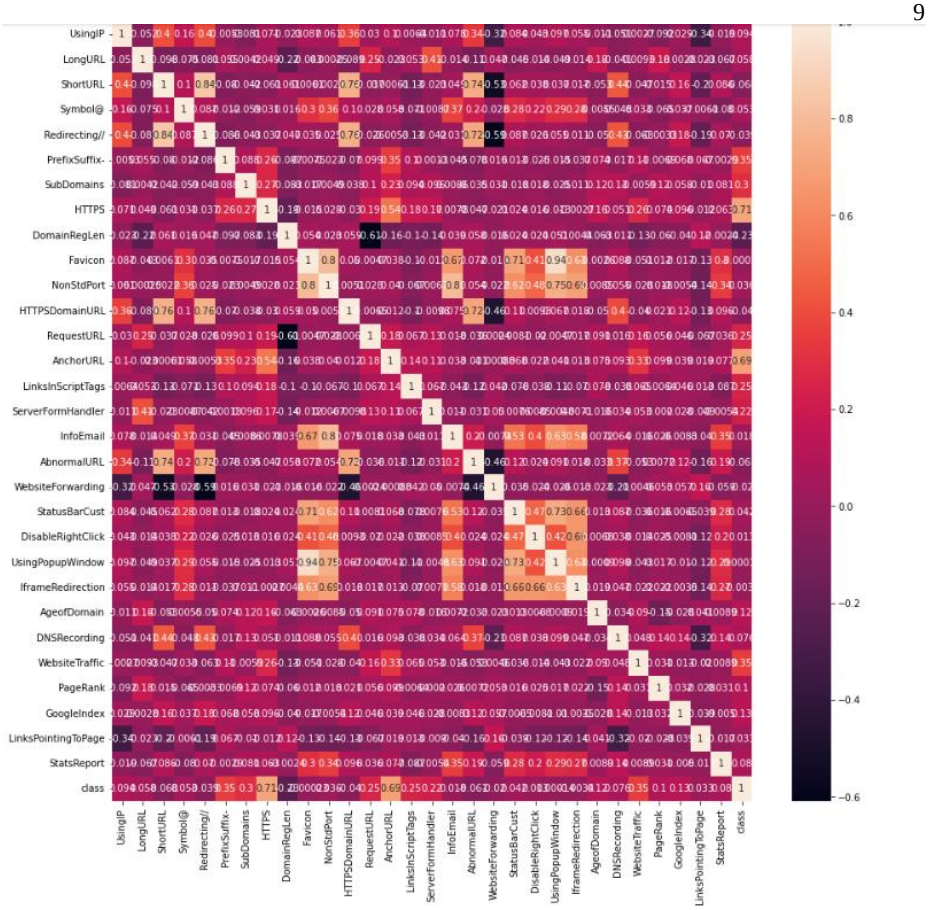


Fig. 6. Heatmap

A heatmap in Fig.6. was also plotted to visualize the positive and negative correlations between the features. We found that the feature HTTPS had the highest positive correlation (0.71) in determining the class of the URL, followed by AnchorURL (0.69). There were 3 features: Favicon, UsingPopUpWindow and IframeRedirection that had zero correlation with respect to the class of the URL.

4.3 Results obtained using VirusTotal API key

In Fig.7, we see outputs from the third method where VirusTotal API Key is used. The total detections are the number of antivirus engines that have analyzed that particular URL. The positive detections are the number of antivirus engines that have classified it as malicious. Even if only one antivirus engine detects the URL as malicious, it is declared dangerous and an analysis report is issued as displayed in Fig.8.

```
Enter the URL to scan (or type 'exit' to quit): https://11011020.web.fc2.com/
The URL https://11011020.web.fc2.com/ is malicious.
Positive detections: 1
Total detections: 65
A detailed report has been saved in the same directory as this script: /content/ht
Scan another URL? (y/n): y
Enter the URL to scan (or type 'exit' to quit): https://028cdxyk.com/
The URL https://028cdxyk.com/ is malicious.
Positive detections: 11
Total detections: 89
A detailed report has been saved in the same directory as this script: /content/ht
Scan another URL? (y/n):
```

Fig. 7. Checking whether website is malicious or not

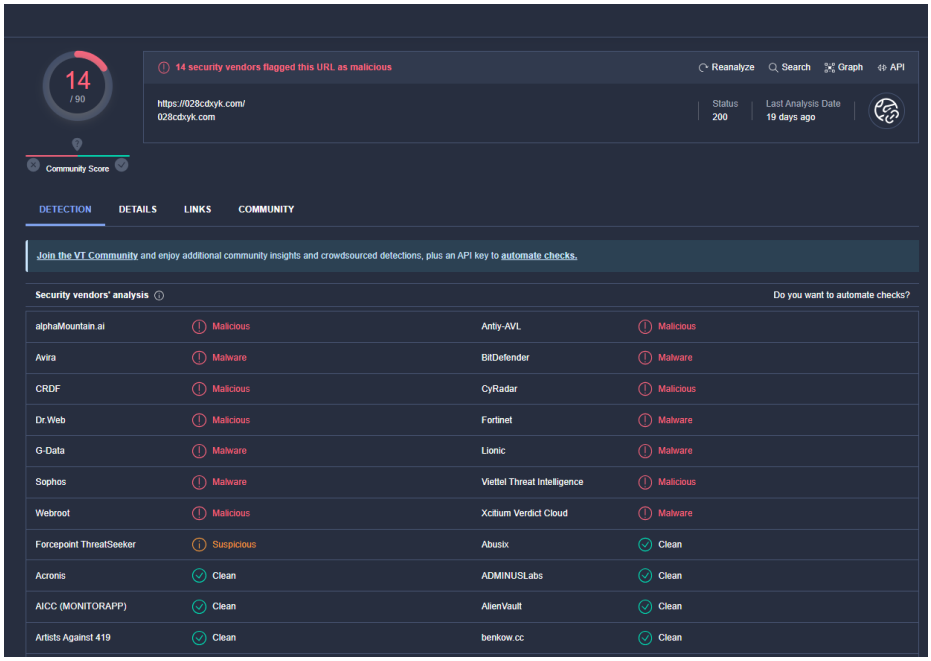


Fig. 8. Analysis Report

5 Conclusion

The main conclusion drawn from this research is to investigate different machine learning models, conduct Data Analysis on the phishing dataset, and comprehend their characteristics. Some of the following observations were concluded :

- Gradient Boosting Classifier gave the best accuracy of 97.4% and also gave the best F1-score of 0.977. Hence, it was the best model to reduce the chance of malicious attachments.
- Naive Bayes Classifier gave the highest precision of 0.997 however it was the worst model as all the other evaluation metrics were low.
- In the exploration of the Phishing dataset, it was revealed that attributes like "AnchorURL", "HTTPS" and "WebsiteTraffic" hold even greater paramount

importance in the classification of a URL as either a phishing URL or a legitimate one.

- "Favicon", "UsingPopUpWindow" and "IframeRedirection" were the features that had zero correlation in the URL classification.
- The results of the self-written code were verified using the VirusTotal API Key. The code proved to be 90% successful in correctly identifying malicious URLs.

6 Statements and Declaration

6.1 Author Contributions

J.M. and A.K. chose the dataset; J.M. wrote the code for segmenting and analyzing URL; J.M. and A.K. implemented the different machine learning models and compared them; J.M. verified the results of the code using VirusTotal API key; J.M. prepared the original draft submission; A.K. conducted a thorough review and made edits to the manuscript. All authors have reviewed and approved the final published version of the manuscript.

6.2 Funding

No external funding was provided to this research.

6.3 Competing interests

The authors state that they have no conflicts of interest

6.4 Acknowledgements

Not applicable

References

1. Sahoo, Doyen, Chenghao Liu, and Steven CH Hoi. "Malicious URL detection using machine learning: A survey." *arXiv preprint arXiv:1701.07179* (2017).

2. Raja, A. Saleem, R. Vinodini, and A. Kavitha. "Lexical features based malicious URL detection using machine learning techniques." Materials Today: Proceedings 47 (2021): 163-166.

3. Begum, Aliya, and Srinivasu Badugu. "A study of malicious url detection using machine learning and heuristic approaches." Advances in Decision Sciences, Image Processing, Security and Computer Vision: International Conference on Emerging Trends in Engineering (ICETE), Vol. 2. Springer International Publishing, (2020)

4. Peng, P., Yang, L., Song, L., Wang, G.: Opening the blackbox of virustotal: Analyzing online phishing scan engines. In: Proceedings of the Internet Measurement Conference, pp. 478–485 (2019)

5. Verma, R., Das, A.: What's in a url: Fast feature extraction and malicious url detection. In: Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics, pp. 55–63 (2017)

6. Mamun, Mohammad Saiful Islam, et al. "Detecting malicious urls using lexical analysis." Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10. Springer International Publishing, (2016)

7. Phadke, Pranav, and Christina Thorpe. "Analysis of API Driven Application to Detect Smishing Attacks." European Conference on Cyber Warfare and Security. Academic Conferences International Limited, (2021)

8. Aonzo, S., Georgiu, G.C., Verderame, L., Merlo, A.: Obfuscapk: An open-source black-box obfuscation tool for android apps. SoftwareX 11, 100403 (2020)

9. Cui, B., He, S., Yao, X., Shi, P.: Malicious url detection with feature extraction based on machine learning. International Journal of High Performance Computing and Networking 12(2), 166–178 (2018)

10. 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. Computer Communications 175, 47–57 (2021)

11. Janet, B., and R. Joshua Arul Kumar. "Malicious URL detection: a comparative study." 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, (2021)

12. Liu, Chunlin, et al. "Finding effective classifier for malicious URL detection." Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences. (2018)

13. Ibrahim, A.A., Ridwan, R.L., Muhammed, M.M., Abdulaziz, R.O., Saheed, G.A.: Comparison of the catboost classifier with other machine learning methods. International Journal of Advanced Computer Science and Applications 11(11) (2020)

14. Weedon, Martyn, Dimitris Tsaptsinos, and James Denholm-Price. "Random forest explorations for URL classification." 2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA). IEEE, (2017)

15. Odeh, Ammar, Ismail Keshta, and Eman Abdelfattah. "Efficient detection of phishing websites using multilayer perceptron." (2020): 22-31.

16. Patil, D.R., Patil, J.B., et al.: Malicious urls detection using decision tree classifiers and majority voting technique. Cybernetics and Information Technologies 18(1), 11–29 (2018)

17. Assegie, T.A.: K-nearest neighbor based url identification model for phishing attack detection. Indian Journal of Artificial Intelligence and Neural Networking 1, 18–21 (2021)

18. Chiramdasu, Rupa, et al. "Malicious url detection using logistic regression." 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS). IEEE, (2021)

19. Subedi, Kul Prasad, Daya Ram Budhathoki, and Dipankar Dasgupta. "Forensic analysis of ransomware families using static and dynamic analysis." 2018 IEEE Security and Privacy Workshops (SPW). IEEE, (2018)

20. Khai, Chang Deng, and Julia Juremi. "BEsafe-Validating URLs and Domains with the aid of Indicator of Compromise." 2023 15th International Conference on Developments in eSystems Engineering (DeSE). IEEE, (2023)