International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8 May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.081 ISSN: 2581-4621

Feature-Driven Machine Learning Approach for Real-Time Malicious and Phishing URL Detection

K. Shalivahana Reddy[1]
M. Kalyan, S. D. Khalil Basha, T. Venugopal, A. Kuladeep Sai, S. Salman
Department of Computer Science and Engineering, K.S.R.M. College of Engineering, Kadapa
Corresponding author1: shali@ksrmce.ac.in

***Abstract*—**Phishing remains the most direct way to obtain sensitive information from an unsuspecting user: attackers persuade victims to hand over usernames, passwords, or bank details rather than breaking any technical control. A phishing URL is a link crafted to promote a scam or fraud; following one can install ransomware, open a credential-harvesting page, or enable other forms of cybercrime. Because attackers exploit human judgement rather than a fixed system flaw, and because blacklists cannot cover domains that have not yet been seen, detection needs to work from properties of the URL itself and needs to stay reliable as attack patterns shift. This paper reports a machine learning approach to phishing URL detection that extracts lexical, host-based, and structural features and trains six classifiers on them: Decision Tree, Logistic Regression, Random Forest, Stochastic Gradient Descent, Naive Bayes, and Support Vector Machine. The classifiers are compared not only on accuracy but on false positive and false negative rates, since the two error types carry different operational costs in this setting. Support Vector Machine achieved the highest accuracy among the six classifiers evaluated (94.0%), followed by Logistic Regression (93.0%) and Naive Bayes (92.7%).

Index Terms*—phishing detection, malicious URL, machine learning, decision tree, random forest, logistic regression, cybersecurity*

I. INTRODUCTION

Phishing is the simplest way to obtain sensitive information from an unsuspecting user. Rather than defeating a technical control, the attacker simply persuades the victim to hand over a username, password, or bank detail. That is why the technique persists no matter how strong the underlying systems become. Domain phishing extends this idea by impersonating a registrar in email, so that a recipient who follows what looks like a routine administrative link ends up surrendering control of a domain.

A phishing URL is a link created to promote a scam, an attack, or a fraud. Clicking one may install ransomware, lead to a page that harvests credentials, or open the way to other cybercrime. The methods attackers use, among them social engineering, phishing, and pharming, target end-to-end technology and human judgement together, and network insecurity is rising in both volume and severity. Security practitioners consequently need detection techniques that are trustworthy and stable rather than purely reactive.

The obvious defence does not scale. Blocking known bad addresses works only for addresses already known to be bad, and a phishing campaign typically registers fresh domains hours before it launches, so a list-based defence is always behind. What is needed is a method that can judge a URL it has never seen, from the properties of the URL itself.

This study applies machine learning to that problem: extracting and analysing features of legitimate and phishing URLs, then training six classifiers, Decision Tree, Logistic Regression, Random Forest, Stochastic Gradient Descent, Naive Bayes, and Support Vector Machine, to distinguish between them. The comparison reports false positive and false negative rates alongside accuracy. This distinction matters in deployment: a false negative admits an attack, while a false positive blocks a legitimate site and teaches users to override the warning.

The remainder of this paper is organised as follows. Section II reviews existing phishing detection approaches. Section III describes the proposed feature set, the classifiers used, and the system

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8 May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.081 ISSN: 2581-4621

architecture. Section IV presents the experimental results and discusses their implications. Section V concludes the paper and outlines directions for future work.

II. RELATED WORK

Phishing detection in the literature falls into four broad families. The list-based approach checks a URL against a maintained whitelist and blacklist, granting access only if the address is on the whitelist and refusing it if it is on the blacklist [3]. This is exact for addresses the list already contains, but static: it cannot classify anything it has not seen before, and a campaign that registers a new domain shortly before launch defeats it by construction.

The heuristic approach derives a pattern from URLs already labelled as phishing and classifies new URLs by how closely they match it [1], [2]. Feature engineering, rather than the choice of learning algorithm on its own, tends to be the step that determines classification accuracy in this approach.

The visual similarity approach takes a server-side rendering of a suspect page and compares it against the legitimate original using image processing. Because a fake page is built to look almost identical to the one it imitates, differences too small for a human viewer to notice are still detectable through pixel- or layout-level comparison.

The content-based approach analyses the text and structure of the page itself. It is informative, but it requires the page to be fetched before a verdict is possible, which is the one action a user should avoid taking on a link they do not trust. The approach adopted in this paper is heuristic in the sense above: it works on properties of the URL string alone, so a classification is available before anything is retrieved from the destination server.

III. PROPOSED METHODOLOGY

A. System Architecture

The proposed detection system has three parts: a set of URL features and behaviours, a machine learning model trained on labelled data, and the infrastructure needed to apply that model at the scale real traffic requires. Fig. 1 shows the detection pipeline. Features are extracted from an incoming URL and passed to the trained model together with a knowledge base of known attack signatures; the model's output then drives a decision to block the address or raise an alert.

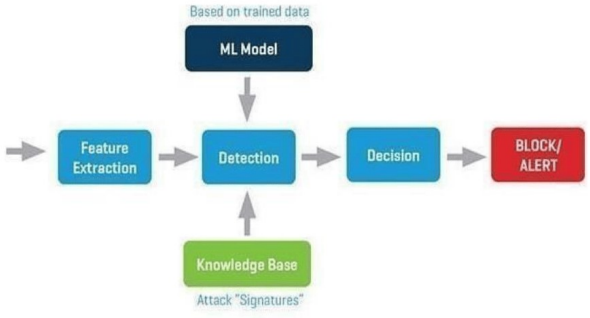

Fig. 1. Detection pipeline of the proposed system.

B. Feature Extraction

The features extracted from a URL fall into three groups. Lexical properties include the length of the URL, the number of dots, whether an IP address appears in place of a hostname, and the presence of characters that rarely appear in legitimate addresses. Host-based properties include the age of the domain and its registrar. Structural properties include the depth of the URL path and the presence of a subdomain that reproduces a well-known brand name to mislead the reader. No single feature is conclusive on its own; a classifier that combines several of them outperforms any one rule applied in isolation.

C. Classification Algorithms

Six classifiers were trained on the extracted features and compared.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8 May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.081 ISSN: 2581-4621

1) Decision Tree:

Decision tree classifiers capture descriptive decision-making knowledge from data and can be built directly from a training set. Given a set S of objects, each belonging to one of the classes C_1 through C_k, construction proceeds recursively: if every object in S belongs to the same class, the tree is a leaf labelled with that class; otherwise a test T with outcomes O_1 through O_n is selected, S is partitioned into subsets S_1 through S_n according to the outcome each object produces, T becomes the root, and the procedure repeats on each subset.

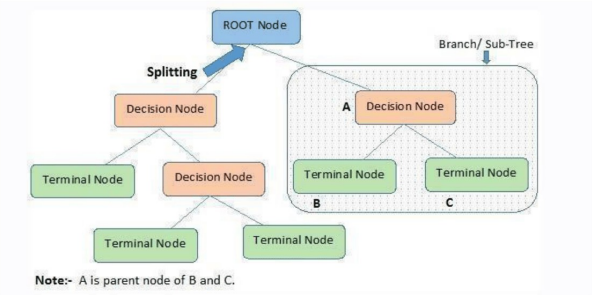

Fig. 2. Structure of a decision tree.

Fig. 2 shows the resulting structure. The root node represents the entire sample and is divided into homogeneous subsets. Splitting divides a node into sub-nodes; a decision node is a sub-node that splits further; a leaf, or terminal node, does not split; pruning removes sub-nodes from a decision node, reversing a split; and a branch, or sub-tree, is a subsection of the whole tree.

2) Logistic Regression:

Logistic regression models the relationship between a categorical dependent variable and a set of independent variables, and it applies directly where the dependent variable takes two values, exactly the phishing-or-legitimate decision made here. It is often preferred to discriminant analysis because it does not require the independent variables to be normally distributed, an assumption that URL-derived features routinely violate.

3) Random Forest:

Random forest builds an ensemble of decision trees, each trained on a bootstrap sample of the data with a random subset of features considered at every split, and combines their predictions by majority vote. Averaging over many de-correlated trees reduces the variance of a single decision tree without much increase in bias, and the resulting ensemble typically generalises better than any individual tree within it.

4) Stochastic Gradient Descent:

The stochastic gradient descent classifier fits a linear model by updating its weights from one training example, or a small mini-batch, at a time rather than computing the gradient over the full dataset at every step. This keeps training inexpensive on large feature sets and makes the classifier easy to refresh as new labelled URLs arrive.

5) Naive Bayes and Support Vector Machine:

Naive Bayes applies Bayes' theorem under the assumption that features are conditionally independent given the class. That assumption is rarely exactly true, but it is often harmless in practice and it makes training extremely fast. The support vector machine finds the hyperplane that maximally separates the two classes in feature space; with a kernel transformation it also handles boundaries that are not linear in the original features.

D. Implementation

The system is implemented in Python and delivered as a web application with two roles. A service provider logs in to upload and browse datasets, train and test the classifiers, view accuracy in tabular and chart form, inspect the distribution of predicted classes, and manage remote users. Remote users register, log in, and submit URLs for classification, with each result recorded against their profile. Separating the two roles keeps model training under administrative control while still letting any registered user query the trained model.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8 May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.081 ISSN: 2581-4621

IV. RESULTS AND DISCUSSION

A. Performance Comparison

All six classifiers were trained and tested on the same labelled dataset of URLs. Table I summarises their accuracy on the held-out test set.

TABLE I
Classifier Accuracy on the Test Set

Classifier	Accuracy (%)
Support Vector Machine	94.00
Logistic Regression	93.00
Naive Bayes	92.73
Decision Tree	91.00
Random Forest	88.00
Stochastic Gradient Descent	87.00

Support Vector Machine achieved the highest accuracy (94.0%), with Logistic Regression (93.0%) and Naive Bayes (92.7%) close behind. Decision Tree (91.0%), Random Forest (88.0%), and the SGD classifier (87.0%) trailed. The margin- and probability-based classifiers outperforming the tree-based ones here suggests the feature set separates phishing from legitimate URLs in a way that is close to linear. That is plausible: several of the strongest lexical indicators, such as URL length, dot count, and the presence of an IP address in place of a hostname, tend to behave close to monotonically with phishing likelihood rather than interacting in the threshold-heavy way that usually favours tree ensembles.

B. Error Analysis

Accuracy alone is a misleading basis for comparison here, because false positives and false negatives are not equally costly. A false negative admits a phishing URL and may cost a user their credentials. A false positive blocks a legitimate site, and repeated false positives train users to click through warnings, which erodes the value of the whole system. A classifier with slightly lower accuracy but a markedly lower false positive rate can therefore be the better choice for deployment, even where, as here, it is not the top performer on accuracy alone.

C. Limitations

Two limitations are worth recording. Features derived from the URL alone cannot detect a phishing page hosted on a compromised but genuinely legitimate domain, where the address itself gives no indication that anything is wrong. And attackers adapt to detection systems: once a feature such as URL length becomes discriminative, campaigns are constructed to avoid it, so a deployed model needs periodic retraining on recent data rather than a single training pass.

V. CONCLUSION AND FUTURE WORK

This paper applied machine learning to the detection of malicious and phishing URLs, extracting lexical, host-based, and structural features and training six classifiers, Decision Tree, Logistic Regression, Random Forest, Stochastic Gradient Descent, Naive Bayes, and Support Vector Machine, for comparison. Classifying from URL features alone allows a verdict before the destination page is retrieved, which is what makes the approach usable as a protective control rather than as after-the-fact analysis. Support Vector Machine gave the best accuracy among the classifiers tested, at 94.0%, with Logistic Regression and Naive Bayes close behind.

Future work should combine URL features with lightweight content signals for cases where the address alone is uninformative, in particular phishing hosted on compromised legitimate domains. Periodic retraining on recent data is necessary because the adversary adapts, and an evaluation protocol that tests on campaigns launched after the training period would measure that adaptation honestly rather than reporting optimistic within-sample figures. Deployment as a browser extension or a mail-gateway filter, where the classification reaches the user at the moment of decision, would turn the model from an analysis tool into an active defence.

DOI:10.46647/ijetms.2024.v08i03.081 ISSN: 2581-4621

REFERENCES

[1] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," Neural Computing and Applications, vol. 25, pp. 443–458, 2014.
[2] S. Marchal, J. François, R. State, and T. Engel, "PhishStorm: detecting phishing with streaming analytics," IEEE Trans. Network and Service Management, vol. 11, no. 4, pp. 458–471, 2014.
[3] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in Proc. ACM SIGKDD, 2009, pp. 1245–1254.
[4] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, no. 1, pp. 81–106, 1986.
[5] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.
[6] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
[7] D. W. Hosmer and S. Lemeshow, Applied Logistic Regression, 2nd ed. New York, NY, USA: Wiley, 2000.
[8] A. K. Jain and B. B. Gupta, "A survey of phishing attack techniques, defence mechanisms and open research challenges," Enterprise Information Systems, vol. 16, no. 4, 2022.
[9] F. Pedregosa et al., "Scikit-learn: machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.