International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Malicious URLs Detection Using Machine Learning Models

Armaanpreet Singh Khehra

Invictus International School, Amritsar

Abstract: ***This research focuses on developing a machine learning - based system for determining malicious links. The main goal is to distinguish between benign and malicious web links using automated classification techniques. To achieve this, we used multiple datasets, including the Malicious URLs Dataset by Manu Siddhartha and the URL Dataset by Antony J. These datasets provide a vast compilation of URLs labeled in different categories, making it easy for effective model training and evaluation. This approach involves training various machine learning algorithms, comprising Random Forest, Gradient Boosting, and Multi - Layer Perceptron models. Feature extraction procedures such as one - hot encoding and URL parsing etc. were put into practice to maximize model's performance. The trained models achieved an accuracy of 85%, and have shown substantial possibilities for real - world cybersecurity applications. This prototype serves as a promising move toward automated URL threat detection, with future work focusing on improving detection efficiency and managing evolving cyber threats.***

Keywords: Malicious URLs, Machine Learning, Cybersecurity, URL Classification, Feature Extraction

1. Introduction

Since the 90s, when the first websites ever were built, the internet has grown exponentially and a large portion of all human activity is nowadays carried online. In parallel to this trend, cyber fraudsters are constantly evolving their skills and methodologies to commit crimes on the internet. Threat actors tend to use malicious URLs to deceive users and launch attacks intended to extract information by mainly tricking inexperienced end users, resulting in compromising the user's system and causing losses of billions of dollars each year [1]. Hence, there is a constant need to develop new tools that ensure a secure online experience for everyone.

2. Background

A **uniform resource locator** (**URL**), colloquially known as an **address** on the Web, is a reference to a resource that specifies its location on a computer network and a mechanism for retrieving it [2]. A malicious URL is a link created with the purpose of promoting scams, attacks, and frauds. When accessed, malicious URLs can download ransomware, open the path for phishing attacks, or cause other forms of cybercrime.

Many people click on links online or in emails without taking into account security considerations beforehand.

People checking email on their phone or laptop are often not protected by URL filtering and other services within the network.

Furthermore, the real URL behind a hyperlink is often not displayed on the main body of the webpage that the end user sees: most people don't hover over links to see where they actually lead, making it easy for bad actors to simply rename the links.

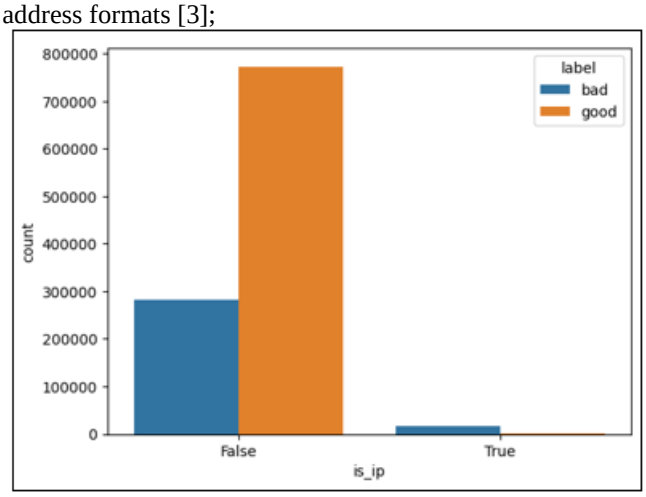
3. Dataset

Two existing datasets from Kaggle [15] [16] are used. They contain 651191 and 411247 malicious and benign urls respectively. The two datasets were standardized through minimal preprocessing and merged into a single tabular format. The resulting dataset was then cleaned to select specific features that we need.

We chose some features describing different characteristics of the URLs that can help determine if a URL is malicious or not. These features are engineered and selected to provide maximum discrimination power with respect to the target category. The following list defines the features we used, and the distribution of each of these features is shown, comparing the two target categories (where necessary, the same distribution is displayed with two different x - axis ranges, to highlight its features):

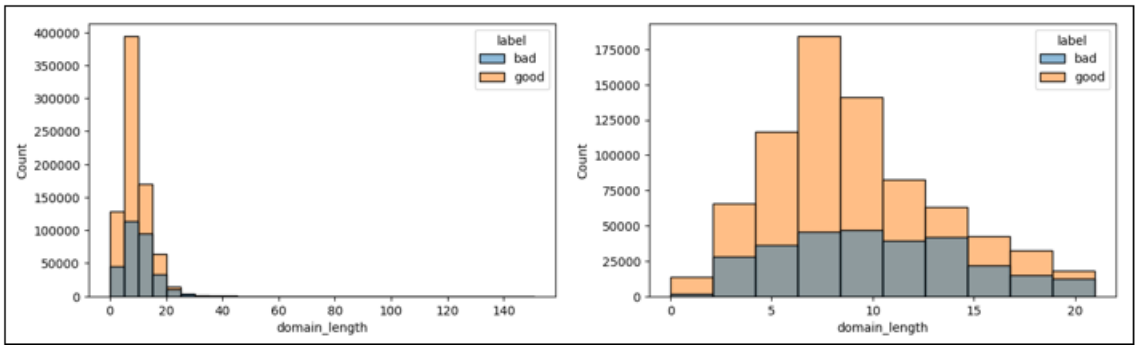
1) Containing an IP address

Regular expressions were used to verify compatibility with IP address formats [3];

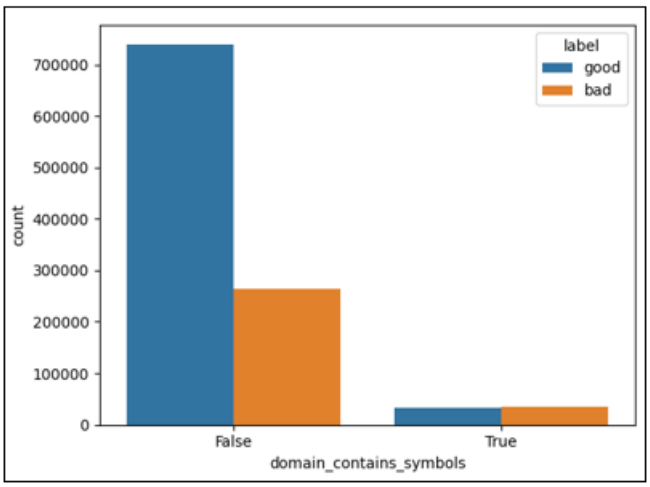


2) URL length

This is simply the count of characters. Threat actors at times use this feature to make the URL less readable;



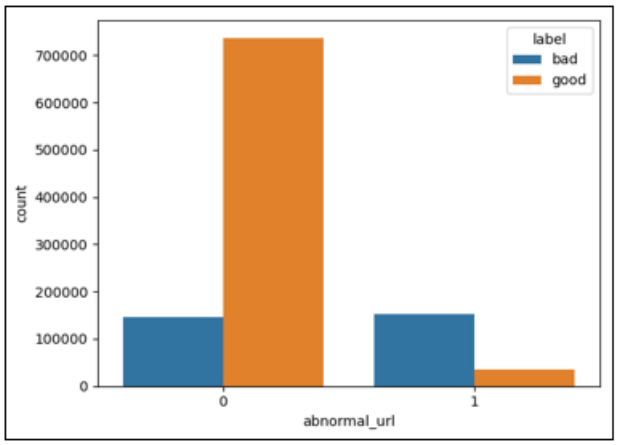
3) Special symbols



Checks for any special characters using regular expressions. While some special characters are normally part of a URL structure, we observed that a high number of those is correlated with the URL being malicious;

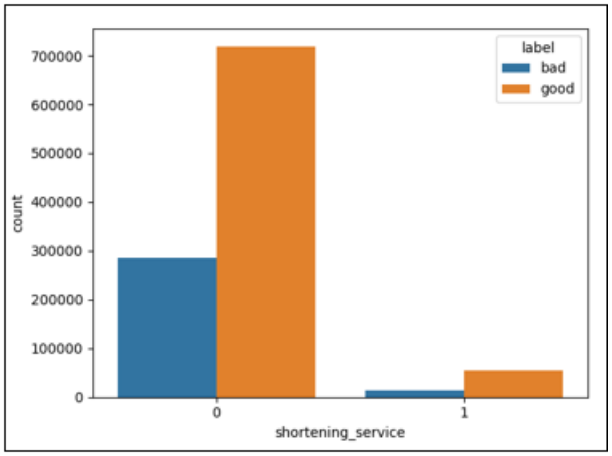
4) Abnormalities

We use URL parsers [4] to identify whether the URL follows the standard structure in terms of domain, subdomain and path: URLs not following the standard format tend to be associated with suspicious activity;



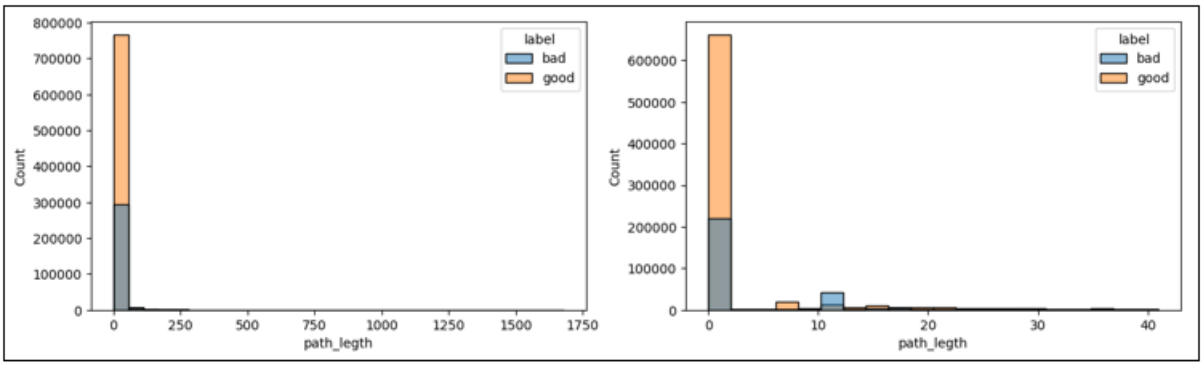
5) Link shorteners

URLs from link shortening services are - according to our exploratory data analysis - often hiding malicious destinations;



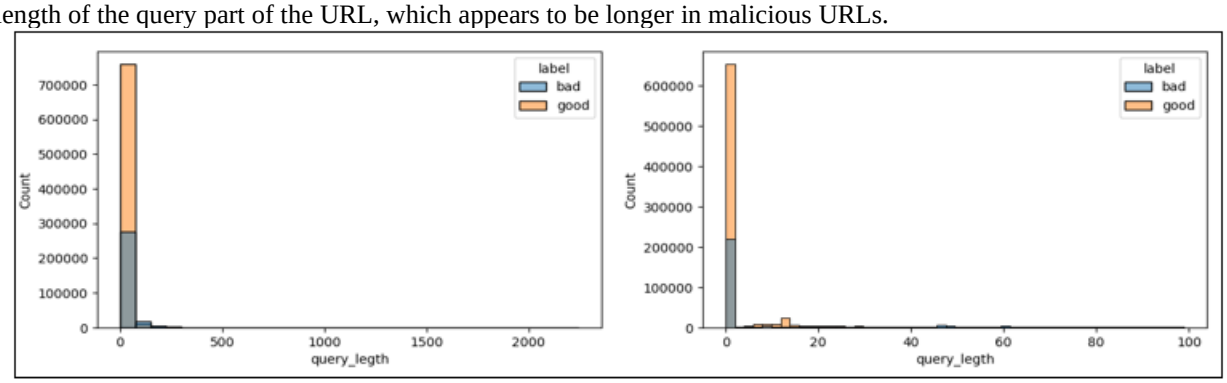
6) Path length

The length of the path part of the URL, which appears to be longer in malicious URLs.



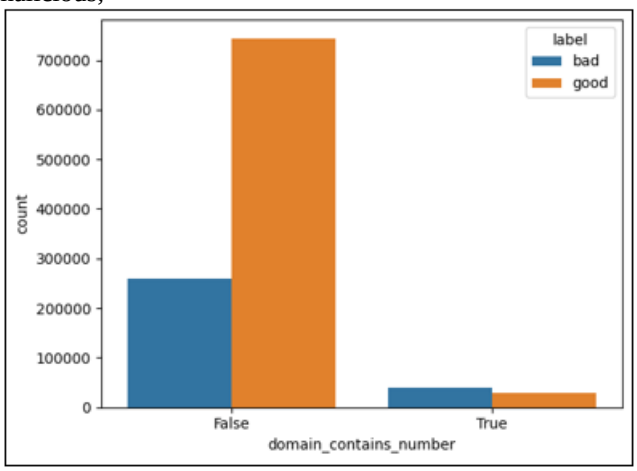
7) Query length

The length of the query part of the URL, which appears to be longer in malicious URLs.



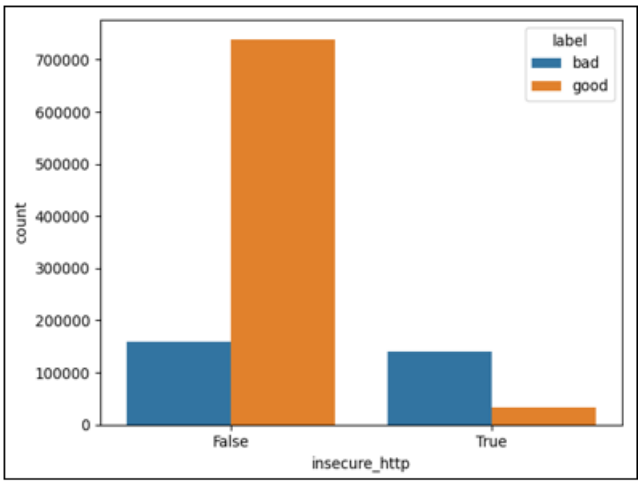
8) Numbers

URL containing many numbers seem to be more often malicious;



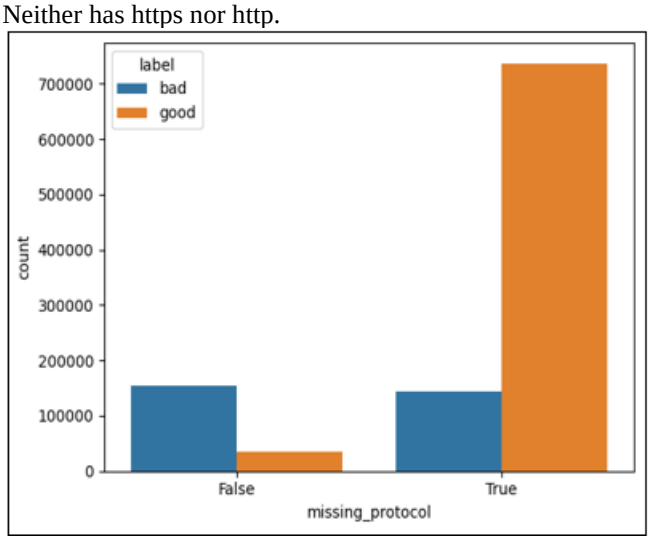
9) Secure HTTP protocol

We check whether the url contains or not the specification to the https protocol.



10) Missing Protocol

Neither has https nor http.

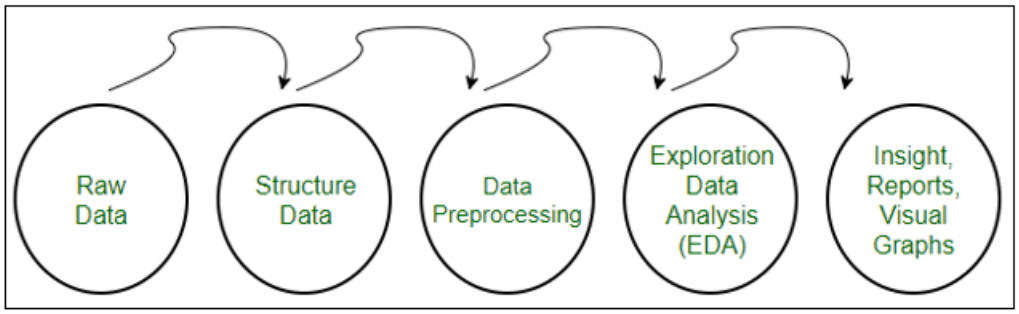


4. Methodology/ Models

After extracting the above mentioned features using regular expressions and basic NLP techniques, we then proceed to implement classification models to categorize the URLs in our dataset.

1) Preprocessing:

Pre - processing refers to the transformations applied to our data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert the raw data into a clean data set [5]. The image below explains pre - processing in Machine Learning.

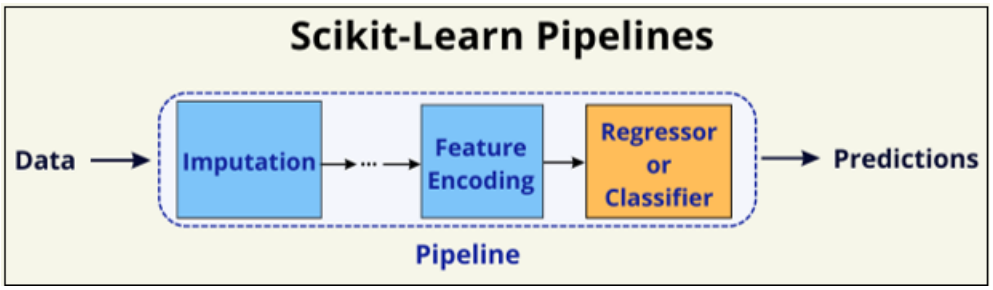


First, both datasets were combined, followed by one - hot encoding of the categorical features [6]. Benign Urls were marked as good and all the others that were earlier labeled as defacement, malware or phishing were marked as bad.

Then we extracted features mentioned above using tldextract [7], re (regular expressions) [8] and urllib. parse [9].

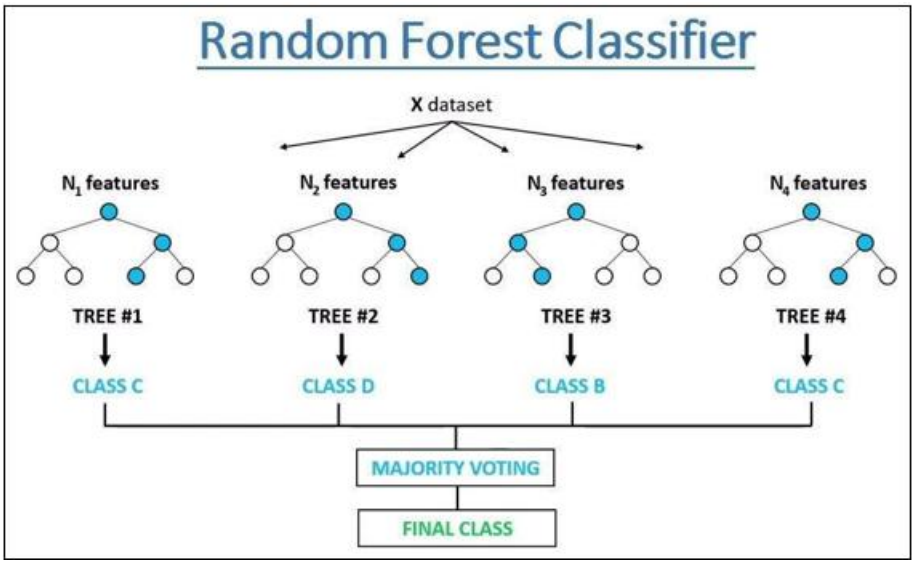
2) Models:

Three supervised machine learning models were used to classify the data: Random Forest, Gradient Boosting and Multilayer Perceptron which were fed by a pipeline. A machine learning pipeline is a series of interconnected data processing and modeling steps designed to automate, standardize and streamline the process of building, training, evaluating and deploying machine learning models [10]. The figure below shows the schematics of a typical machine learning pipeline.



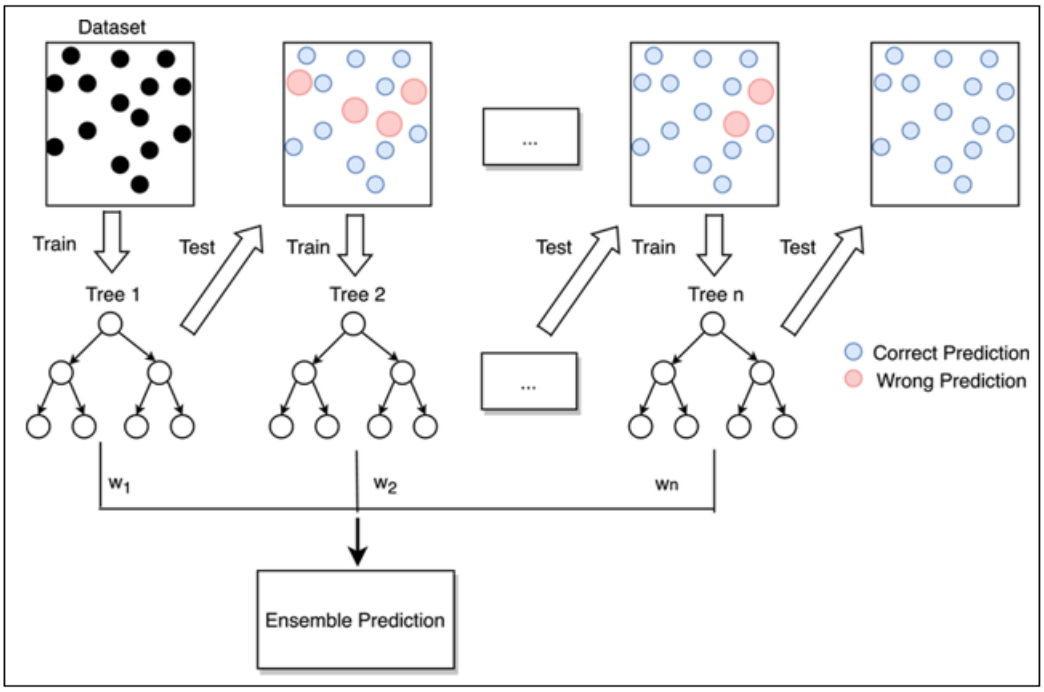
a) Random Forest Classifier

Random forest classifier trains a set of decision trees from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction [11]. The figure below represents the composition of a generic Random Forest Classifier.



b) Gradient Boosting Classifier

Gradient Boosting Classifier is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross - entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met [12]. See the image below for a visual representation of this architecture.

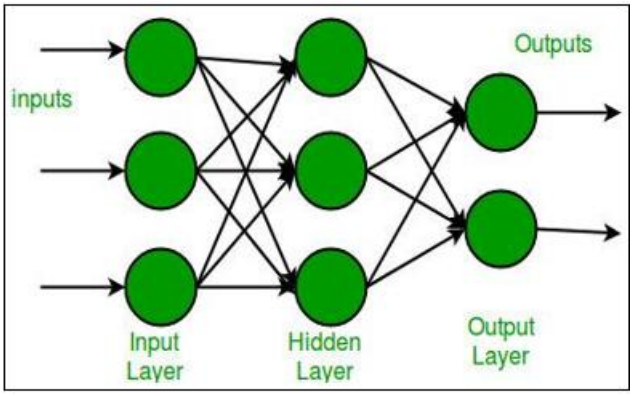


c) Multilayer Perceptron

A ***Multi - Layer Perceptron (MLP)*** is a type of neural network that consists of fully connected dense layers that transform input data from one dimension to another. It is called "multi - layer" because it contains an input layer, one or more hidden layers, and an output layer. The purpose of an MLP is to model complex relationships between inputs and outputs, making it a powerful tool for various machine learning tasks [13].

The key components of Multi - Layer Perceptron includes:

- **Input Layer:** Each neuron (or node) in this layer corresponds to an input feature. For instance, if you have three input features, the input layer will have three neurons.
- **Hidden Layers:** An MLP can have any number of hidden layers, with each layer containing any number of nodes. These layers process the information received from the input layer.
- **Output Layer:** The output layer generates the final prediction or result. If there are multiple outputs, the output layer will have a corresponding number of neurons.

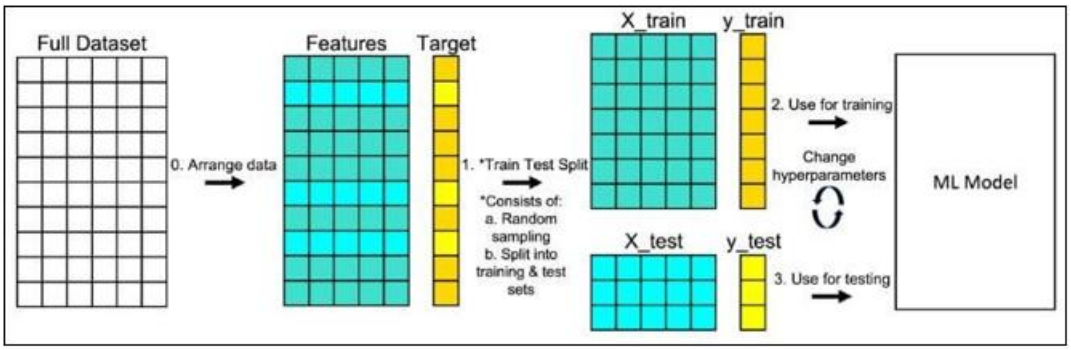


The image above represents the architecture of a generic MLP.

Every connection in the diagram is a representation of the fully connected nature of an MLP. This means that every node in one layer connects to every node in the next layer. As the data moves through the network, each layer transforms it until the final output is generated in the output layer.

3) Train - Test Split:

In our study, we divided the dataset into two subsets: a training set and a testing set. The **training set** (80% of the dataset) was used to train the models, allowing them to learn patterns and relationships within the data. Once the models were trained, we evaluated their performance using the **testing set (**20% of dataset) —data that the models had not previously encountered. The illustration below visualizes Train - Test Split.



This process ensures an **objective assessment** of the models' ability to generalize to new, unseen data, rather than just memorizing the training data. Additionally, splitting the dataset in this way helps us monitor and prevent **overtraining** (or overfitting), where a model becomes too tailored to the training data and performs poorly on new data. By comparing performance on both subsets, we can ensure that our models are robust and capable of making reliable predictions.

4) Hyperparameters Tuning

Hyperparameter tuning is essential for optimizing the performance of machine learning models. This technique allowed us to find the best combination of hyperparameters that maximize model performance. For the Random Forest Classifier, we tuned `n_estimators`. For Gradient Boosting Classifier, we tuned parameters such as `learning_rate`, `n_estimators`, and `verbose`. For the Multilayer Perceptron Classifier, we tuned parameters including 'hidden_layer_sizes', 'verbose', 'learning_rate', 'learning_rate_init' and 'early_stopping'.

5. Results

For the results to be comprehensible, one should know about estimation of performances given below

Estimation of performances

The next step is to evaluate the performance of our classifiers. To be able to correctly evaluate our models, we make use of confusion matrix as shown in and use F1score, accuracy, precision and recall as the evaluation metrics.

F1 Score: It is a function of precision and recall, calculated using the average of precision and recall.
F1 = 2* (Precision*Recall) / (Precision+Recall)

Accuracy: This is defined as the overall success rate of the URL prediction technique.
Accuracy = (TP+TN) / (TP+TN+FP+FN)

Precision: This is the ratio of the positive predictions of URLs that are correctly classified.
Precision = TP/ (TP+FP)

Recall: Can be seen as out of all the positive classes (URLs), how much was actually correctly predicted
Recall = TP/ (TP+FN)

TP: number of true positives, actual malicious URLs classified correctly

TN: number of true negatives, actual benign URLs classified correctly

FP: number of false positives (error 1), benign URLs classified as malicious

FN: number of false negatives (error 2) malicious URLs classified as benign

The table below summarizes the performances that we obtained, in terms of precision, accuracy and recall for the three different models that we have implemented.

Name of the Model	Precision	Recall	Accuracy	F1 - score
Random Forest Classifier	0.88	0.56	0.85	0.68
Gradient Boosting Classifier	0.86	0.54	0.85	0.67
Multilayer Perceptron Classifier	0.87	0.53	0.85	0.66

6. Conclusions

This study tackles the pervasive cybersecurity issue of threat actors circumventing security measures and using URLs to initiate harmful attacks on unwary individuals. The present project uses machine learning techniques to identify malicious URLs reliably in order to warn the end user, reducing the attack surface of such frauds [14]. While the three distinct models that we have tested yielded similar performances, we obtain the best precision and recall out of the random forest classification algorithm. This finding is likely correlated with the larger training times that would be required for the other two algorithms to catch up and potentially surpass the performances of the former. Overall, these results are promising and call for an extension of this project to bring such algorithms to a deployable product, for example in the form of a web browser extension.

In terms of next steps, several considerations could enhance the project and improve its outcomes. While the model achieved a recall of 56%—a reasonable starting point—it is not ideal. A recall at this level indicates that a significant portion of malicious URLs could still bypass detection, posing potential cybersecurity risks. To address this limitation, we plan to introduce **additional features** that could provide more nuanced insights into the nature of URLs, such as analyzing lexical patterns, examining URL metadata, and incorporating behavior - based features (e. g., tracking redirection behavior).

Moreover, training on a **larger and more diverse dataset** would likely improve the model's ability to generalize. This could involve gathering data from various domains and

geographic regions to better capture different types of malicious URL tactics.

We also aim to explore and compare other classification models, such as ensemble methods, neural networks, or deep learning approaches, to see if they yield better performance metrics, especially in terms of recall and precision.

Finally, an ideal outcome for this project would be to develop a robust and scalable URL classification system that could be integrated into cybersecurity platforms, offering real - time malicious URL detection. Along the way, we would continue to monitor the limitations of the current approach, iteratively improving the model to reduce false negatives without impacting false positives.

Acknowledgments
Special thanks to Manu Siddhartha [15] and Antonyj [16] for providing the datasets.

References

[1] Deloitte, 2024, "Deepfake banking and AI fraud risk, " Deloitte Insights, https: //www2. deloitte. com/us/en/insights/industry/financial - services/financial - services - industry - predictions/2024/deepfake - banking - fraud - risk - on - the - rise. html.
[2] Wikipedia contributors, "URL, " *Wikipedia, The Free Encyclopedia*, Jun.20, 2024. [Online]. Available: https: //en. wikipedia. org/w/index. php?title=URL&oldid=1230124093
[3] IBM, 2024, "IPv4 and IPv6 Address Formats, " IBM Documentation, https: //www.ibm. com/docs/en/ts3500 - tape - library?topic=functionality - ipv4 - ipv6 - address - formats.
[4] Python Software Foundation, 2024, "urllib. parse — Parse URLs into components, " Python 3 Documentation, https: //docs. python. org/3/library/urllib. parse. html.
[5] Kotsiantis, Sotiris & Kanellopoulos, Dimitris & Pintelas, P. . (2006). Data Preprocessing for Supervised Learning. International Journal of Computer Science.1.111 - 117.
[6] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
[7] PyPI, "tldextract, " [Online]. Available: https: //pypi. org/project/tldextract/.
[8] Python Software Foundation, "re — Regular expression operations, " *Python Documentation*, [Online]. Available: https: //docs. python. org/3/library/re. html.
[9] Python Software Foundation, "urllib. parse — Parse URLs into components, " *Python Documentation*, [Online]. Available: https: //docs. python. org/3/library/urllib. parse. html.
[10] IBM, "Machine Learning Pipeline, " [Online]. Available: https: //www.ibm. com/think/topics/machine - learning - pipeline.
[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay.2011. Scikit - learn: Machine Learning in Python. J. Mach. Learn. Res.12, null (2/1/2011), 2825–2830.
[12] Boehmke, Bradley; Greenwell, Brandon (2019). "Gradient Boosting". Hands - On Machine Learning with R. Chapman & Hall. pp.221–245. ISBN 978 - 1 - 138 - 49568 - 5.
[13] Haykin, S. (1994). Neural networks: a comprehensive foundation. Prentice Hall PTR.
[14] R. A. Kemmerer, "Cybersecurity, " 25th International Conference on Software Engineering, 2003. Proceedings., Portland, OR, USA, 2003, pp.705 - 715
[15] S. Axn, "Malicious URLs Dataset, " *Kaggle*, [Online]. Available: https: //www.kaggle. com/datasets/sid321axn/malicious - urls - dataset.
[16] A. J. Antony, "URL Dataset, " *Kaggle*, [Online]. Available: https: //www.kaggle. com/datasets/antonyj453/urldataset/data.