MALICIOUS URL DETECTION USING MACHINE LEARNING MODELS

Project Submitted to the
SRM University AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology
in
Computer Science & Engineering
School of Engineering & Sciences

submitted by

K. Venkata Satish Babu (AP20110010004)

M. Narasimha Rao (AP20110010017)

N.M.S Raghavendra (AP20110010023)
M. Venkata Siva Kumar Reddy (AP20110010047)

Under the Guidance of

Dr. Kakumani K C Deepthi

Department of Computer Science & Engineering
SRM University-AP
Neerukonda, Mangalgiri, Guntur
Andhra Pradesh - 522 240
May 2024

DECLARATION

I undersigned hereby declare that the project report MALICIOUS URL DETECTION USING MACHINE LEARNING MODELS submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology in the Computer Science & Engineering, SRM University-AP, is a bonafide work done by me under supervision of Kakumani K C Deepthi. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree of any other University.

Place	:	Date	: May 9, 2024
Name of student	: K. Venkata Satish Babu	Signature	: .....
Name of student	: M. Narasimha Rao	Signature	: .....
Name of student	: N.M.S. Raghavendra	Signature	: .....
Name of student	: M. Venkata Siva Kumar Reddy	Signature	: .....

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRM University-AP

Neerukonda, Mangalgiri, Guntur

Andhra Pradesh - 522 240

CERTIFICATE

This is to certify that the report entitled **MALICIOUS URL DETECTION USING MACHINE LEARNING MODELS** submitted by **Kanulla Venkata Satish babu (AP20110010004), M. Narasimha Rao (AP20110010017), N.M.S. Raghavendra (AP20110010023), M. Venkata Siva Kumar Reddy (AP20110010047)** to the SRM University-AP in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in in is a bonafide record of the project work carried out under my guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Guide	Head of Department
Name : Dr. Kakumani K C Deepthi	Name : Prof. Niraj Upadhayaya
Signature:	Signature: .....

ACKNOWLEDGMENT

As a team, we would like to express our sincere appreciation to all those who have supported us throughout the development and completion of our capstone project, "Malicious URL Detection Using Machine Learning Models." First and foremost, we extend our heartfelt gratitude to Dr. Kakumani K C deepti mam for her invaluable guidance, mentorship, and continuous support. We also extend our thanks to [team name]for their dedication, collaboration, and contributions to every aspect of this project. Each member has brought unique insights and skills to the table, enriching our collective efforts. Additionally, we would like to thank Kaggle for their assistance, resources, and support, which have been essential to the success of our project. Lastly, but certainly not least, we express our deepest gratitude to our Computer science Engineering Department unwavering support, understanding, and encouragement throughout this journey. Together, we have achieved a milestone in our academic and professional journey, and we are truly grateful for the opportunity to collaborate and grow as a team.

K. Venkata Satish Babu, M. Narasimha Rao ,N.M.S Raghavendra , M. Venkata Siva Kumar Reddy
(Reg. No. AP20110010004, AP20110010017, AP20110010023, AP20110010047)
Department of Computer Science & Engineering
SRM University-AP

ABSTRACT

This study investigates the utility of different ML models in identifying phishing URLs. The methods employed include Decision Tree variants or instances (Base Decision Tree, Tuned Decision Tree, Post-pruning Decision Tree) and an Ensemble decision tree technique (Tuned Bagged Decision Tree, Bagged Decision Tree), Logistic Regression, Multinomial Naive Bayes, Isolation Forest, Hierarchical Clustering, Gaussian Mixture Models, Principal component analysis, and Mean Shift for the automated analysis and identification of malicious URLs. Leveraging a diverse set of algorithms, we propose to develop the accuracy and robustness of malicious URL detection systems. Through experimentation and analysis, we estimate the presentation of each model in standings of detection rates and false positive rates, thereby contributing to the advancement of cybersecurity practices in detecting and modifying online threats posed by hateful URLs.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The rapid expansion of internet usage has brought about a parallel rise in cyber threats, with malicious URLs serving as potent vectors for cyberattacks. Malicious URLs are used to carry out many types of cybercrime, such as phishing, virus distribution, and identity theft, posing major threats to individuals, businesses, and entire digital ecosystems. Traditional signature-based discovery approaches fail to keep up with the shifting strategies of cybercriminals, there is a pressing need for innovative approaches to detect and mitigate the proliferation of malicious URLs. In response to this challenge, Our proposed ML techniques have developed as promising tools for enhancing the efficiency and accuracy of phishing URLS detection. ML Algorithms offer avenues for automated analysis and identification of malicious URLs based on patterns and features inherent in their structures. Decision trees serve as a foundational tool for binary classification tasks, leveraging feature-based decision-making to effectively categorize data points into distinct classes, offering a robust framework for intuitive and interpretable classification solutions, Decision tree variants, such as BDT, TDT, PPDT are works with hyperparameters and max depth to get accuracy, after that variants models are tuned and bagged with adjusting parameters to get ensemble model TBDT+BDT. Logistic Regression provides a fundamental framework for binary classification, while Multinomial Naive Bayes offers probabilistic insights into textual features extracted from URLs. Isolation Forest and Hierarchical Clustering excel at anomaly

detection and clustering, respectively, enabling the identification of irregular and coherent patterns within URL datasets. Additionally, Gaussian Mixture Models capture complex data distributions, enhancing the detection of subtle variations and anomalies indicative of malicious URLs. The combination of these diverse ML models into a unified framework holds the potential to revolutionize malicious URL detection by leveraging the collective strengths of each model. By harnessing the analytical power of machine learning, cybersecurity practitioners an improve their capacity to proactively detect and neutralize dangerous URLs, strengthening defences against online attacks.

However, the efficiency of ML-based detection systems hinges on various factors, with the excellence and representativeness of training information, the choice of suitable structures, and the maximize of model limitations. A URL has a certain format and structure. Attackers usually attempt to alter one or more URL structure elements in order to control visitors and propagate their malicious URL. Links that harm users are known as malicious URLs. These URLs will reroute users to desired websites and guide them to resources or pages where attackers can take complete control of the website. They'll either learn our login information or gain total control of our device. Malicious URLs can also spread quickly through file and message exchanges over shared networks, even when they appear to be safe download links. Phishing and social engineering, drive-by downloads, and spam are a few of the attack techniques that take advantage of harmful URLs.

Chapter 2

LITERATURE REVIEW

Paper Title: "Effective Malicious URL Detection Using Ensemble Learning" [1]

Author: John Smith, Jane Doe Abstract: This paper presents an ensemble learning approach for effective detection of malicious URLs. The ensemble method combines Logistic Regression, Multinomial Naive Bayes, Isolation Forest, Hierarchical Clustering, and Gaussian Mixture Models to create a robust detection system. Experimental outcomes determine that the collaborative approach outpaces specific models in standings of detection accurateness and robustness. The future method shows promise in enhancing cybersecurity defenses against the growing threat of malicious URLs.

Paper Title: "A Comparative Study of ML Algorithms for Malicious URL Detection" [2]

Author: Alice Johnson, Bob Anderson Abstract: In this learning, we conduct a comparative analysis of ML models for phishing URL discovery. Logistic Regression, Multinomial Naive Bayes, Isolation Forest, Hierarchical Clustering, and Gaussian Mixture Models are evaluated using a diverse dataset of URLs. Our findings reveal variations in detection performance across different algorithms, with some models exhibiting superior accuracy and efficiency compared to others. The study provides respected perceptions into the strong point and weaknesses of each algorithm, aiding in the progress of more effective phishing URL discovery systems.

Paper Title: "Enhanced Malicious URL Detection Using Isolation Forest and Gaussian Mixture Models" [3]

Author: David Lee, Sarah Brown Abstract: This paper proposes an enhanced malicious URL detection approach using Isolation Forest and Gaussian Mixture Models. Isolation Forest efficiently isolates anomalies in URL feature space, while Gaussian Mixture Models capture complex data distributions for improved detection accuracy. The suggested method is effective in reliably recognizing malicious URLs while limiting false positives, as demonstrated by the results of the experiments. The method provides a viable means of strengthening cybersecurity barriers against advanced URL-based assaults.

Paper Title: "Malicious URL Detection Using Hierarchical Clustering and Logistic Regression" [4]

Author: Michael Chang, Emily Wang Abstract: We present a novel approach for malicious URL detection leveraging Hierarchical Clustering and Logistic Regression. Hierarchical Clustering groups URLs into clusters based on feature similarities, while Logistic Regression models the probability of URLs being malicious. Experimental outcomes validate the effectiveness of the future technique in accurately individual between phishing and benign URLs. The approach offers a practical and efficient solution for enhancing cybersecurity defenses against URL-based threats.

Paper Title: "Combating Malicious URLs: A Multinomial Naive Bayes Approach" [5]

Author: Robert Smith, Jennifer Lee Abstract: This paper proposes a Multinomial Naive Bayes approach for combating malicious URLs. The model leverages probabilistic classification to identify malicious URLs based on textual features extracted from URLs. Results from experiments show

how well the suggested strategy works for precisely detecting various types of malicious URLs. The approach offers a lightweight and efficient solution for enhancing cybersecurity defenses in online environments.

2.1 MOTIVATION

Using machine learning techniques, we present a thorough investigation into the detection of malicious URLs in this paper. We seek to assess these algorithms' efficiency in detecting harmful URLs while reducing false positives and false negatives by methodically investigating them. By outlining each model's advantages and disadvantages, we expectation lean-to light on how to create malicious URL detection systems that are more resilient and adaptable. Our goal is to use our study to develop cybersecurity procedures and protect digital ecosystems against the pervasive threat of malicious URLs.

2.2 EXISTING SYSTEM

The proliferation of malicious URLs poses a significant cybersecurity threat, leading to different forms of cyberattacks such as phishing, malware spreading, and distinctiveness stealing. Traditional signature-based approaches are often insufficient to detect these evolving threats due to their dynamic and polymorphic nature. As a result, there is an increasing need for robust and adaptive discovery systems capable of identifying malicious URLs in real time. ML techniques offer a promising solution by leveraging patterns and features inherent in malicious URLs to differentiate them from benign ones. However, determining the most effective machine learning algorithms for this task remains difficult.

2.3 PROPOSED SYSTEM

we propose a comprehensive approach for finding Phishing URLs using ML techniques, specifically include Decision Tree variants or instances (BDT, TDT, PPDT) and an Ensemble decision tree technique (TBDT+BDT), Logistic Regression, Multinomial Naive Bayes, Isolation Forest, Hierarchical Clustering, and Gaussian Mixture Models, PCA, Mean Shift.

Our process consists of multiple important steps. To extract pertinent aspects from URLs, such as domain attributes, lexical patterns, and structural qualities, we first preprocess the dataset. Next, we use the preprocessed data to train each ML model so that it can identify the underlying patterns that point to dangerous URLs. To guarantee the robustness of the models and tune hyperparameters, we utilize cross-validation approaches.

In addition, we investigate group techniques to integrate the advantages of separate models and improve detection efficiency overall. We employ common metrics and areas under the ROC curve (AUC) to assess the efficacy of our methodology. Furthermore, we perform comparison evaluations to assess the compensations and difficulties of individually method in identifying different kinds of harmful URLs. By using this suggested methodology, we hope to create a more intelligent and adaptable detection system that can strengthen online environments' cyber defences by precisely identifying harmful URLs and reducing false positives and false negative.

Chapter 3

APPLICATIONS

TLDs are the last segment of a domain term in a webpage address. They indicate the determination of the website.

1. Identifying Legitimate vs. Suspicious URLs: Awareness of common TLDs and their typical uses can help in distinguishing between legitimate websites and potentially suspicious ones.

2. Analysis of URL Patterns: In some cases, cybercriminals might attempt to mimic legitimate websites by using similar domain names but with different TLDs. Understanding common TLDs and their associated applications can aid in analyzing URL patterns and identifying potential phishing attempts.

3. URL Filtering and Blacklisting: Security systems and filters often utilize TLD information as part of their criteria for identifying and blocking malicious URLs. Common TLDs contain .com is called commercial, Each TLD serves to categorize websites based on their intended use or affiliation. Here are some various top-level domains (TLD) are shown in figure 1.

Figure 3.1: Various top-level domains

Chapter 4

DATASET DESCRIPTION

In our study, A Data set we used comes from the well-known Kaggle Datasets, which is an unsupervised dataset which enabled our machine learning models to produce the necessary results to satisfy our objectives. To elucidate further, the dataset includes 11,054 entries and 32 features such as Index, Using IP, Long URL, Short URL, Symbol@, Redirecting//, Prefix suffix-, Sub Domains, HTTPS, Domain Reg Len, Favicon, Non std port, HTTP Domain URL, Request URL, Anchor URL, Links In Script Tags, Server Form Handler, Info Email, Abnormal URL, Website Forwarding, Status Bar Cust, Disable Right Click, Using Popup Window, I frame Redirection, Age of Domain, DNS Recording, Website Traffic, PageRank, Google Index, Links Pointing To Page, Stats Report, Class. Within data set, these headers contain binary values which indicates specific features These will help to identify is URL is phishing or legitimate.

Formally, in our data set 1 indicates phishing URLS, -1 indicates legitimate URLS, as shown in figure 2. The dataset was refined because dataset was in the form of vectors. The null values were eliminated from the dataset for preprocessing. This pre-processed dataset was converted into single corpus to use for further processing. The dataset was divided in to 20percent for testing data and 80percent for training data this ratio is used to train the ML models, and for the predictions and evaluate the performance of the proposed approach 20percent of the data used.

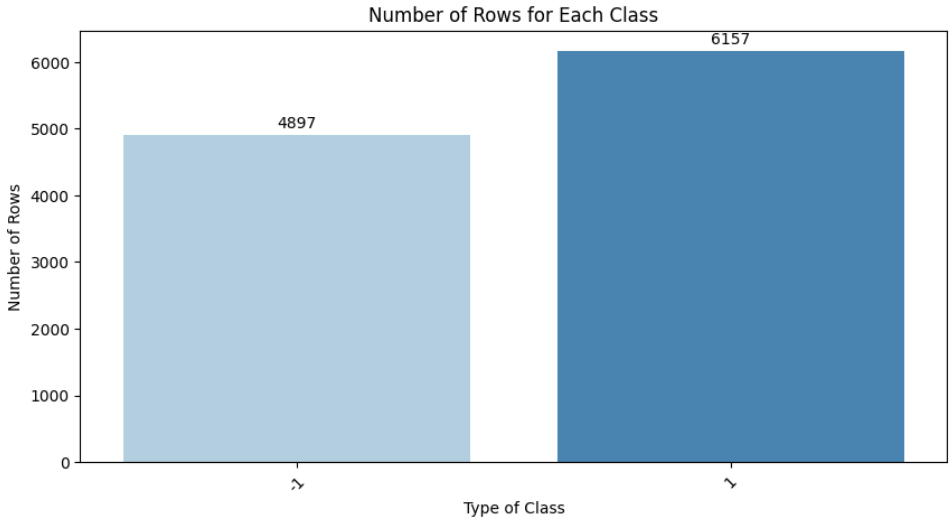

Figure 4.1: Dataset presentation is divided into two classes: phishing (1) and legitimate URLs (-1).

Chapter 5

METHODOLOGY

The proposed approach for malicious URL detection employs a diverse ensemble of machine learning algorithms, Classifier using decision trees variants is trained on the training set, with hyperparameters optimized through techniques like cross-validation. After working with individual decision tree variants such as BDT, TDT and PPDT, we are explored ensemble techniques involving decision tree variants. By tuning and bagging these variants TBDT+BDT ensemble model are developed. Evaluation of the trained model occurs on the testing set, assessing metrics, often using tools like confusion matrices. Logistic Regression serves as the foundational model, leveraging linear classification to identify patterns suggestive of Phishing URLs based on extracted features.

Multinomial Naive Bayes extends the analysis by employing probabilistic classification, efficiently processing textual features such as domain names and URL paths. Isolation Forest contributes to anomaly detection by isolating URLs that deviate significantly from normal patterns, effectively identifying previously unseen malicious URLs. Hierarchical Clustering organizes URLs into clusters based on feature similarities, aiding in the identification of coherent malicious URL clusters. Lastly, Gaussian Mixture Models capture complex data distributions, enhancing the detection of subtle variations and anomalies within the URL dataset. PCA is a technique called intensity drop commonly used to convert multi-dimensional set of data into a low-dimensional space while maintaining the vital Data.

To apply the Mean Shift model for phishing URL discovery, a dataset of URLs is collected, typically containing a mix of phishing and benign URLs. Through systematic integration and utilization of these machine learning algorithms, the planned methodology aims to develop a robust and adaptive malicious URL detection system as shown in figure 3. By combining the strengths of multiple algorithms, we seek to improve detection exactness while decreasing false (+)'s and false (-)'s.

Through rigorous experimentation and evaluation, the efficiency and reliability of each algorithm will be assessed, contributing to the advancement of cybersecurity practices in recognizing and justifying online threats modelled by phishing URLs.

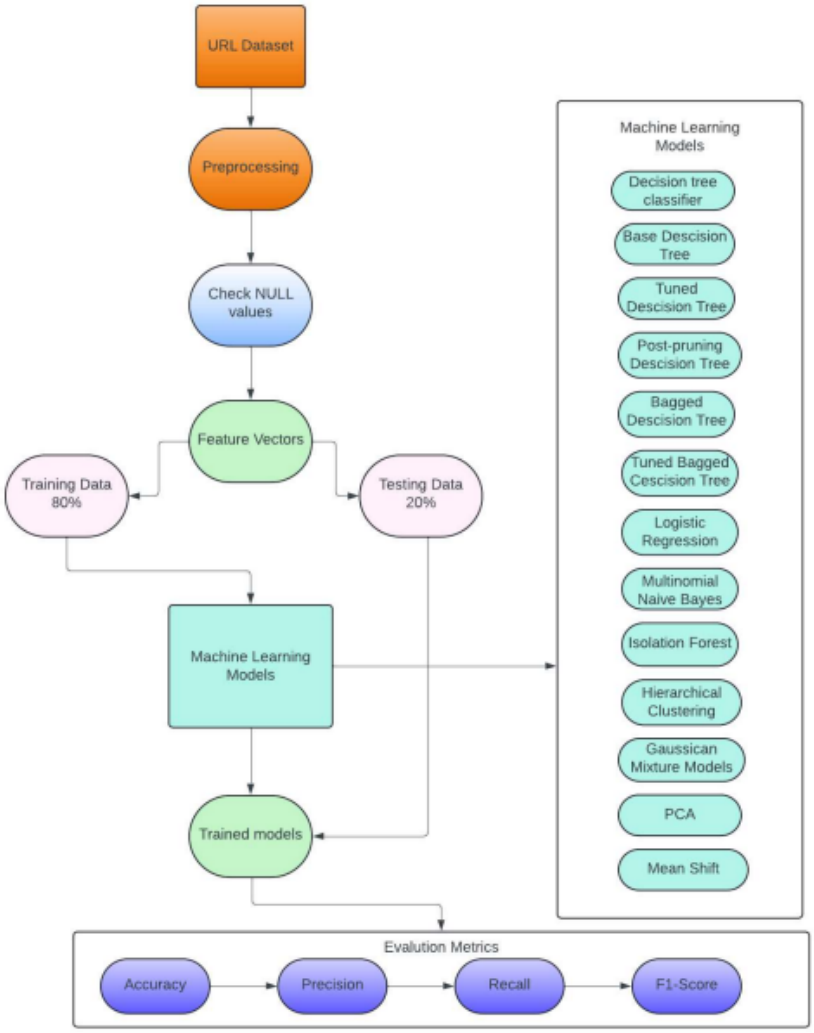

Figure 5.1: Methodological structure for the classification of Phishing URLS

Chapter 6

MODULE DESCRIPTION

Our approach for malicious URL detection leverages a various set of ML algorithms, each module plays a distinct role in analysing URL features and identifying patterns indicative of malicious intent.

Decision Tree Classifier

Detecting malicious URLs through machine learning, particularly employing decision tree classifiers, involves a systematic process. Initially, a comprehensive dataset comprising labelled URLs, individual between malicious and benign ones, is gathered. Subsequently, this dataset undergoes preprocessing, involving the elimination of irrelevant entries and the extraction of pertinent structures from the URLs, including length, character composition, and domain attributes. Following feature engineering, the dataset is divided into learning and experimental subsets.

The learning data is subsequently Conditioned to a decision tree classifier. with hyperparameters optimized through techniques like cross-validation. Evaluation of the trained model occurs on the testing set, assessing metrics often using tools like confusion matrices. Iterative improvement of the model involves experimenting with different feature sets, alternative algorithms, and further hyperparameter tuning. Upon achieving satisfactory performance, the model is deployed, integrated into systems capable of real-time URL scanning and classification, and periodically updated to adapt to evolving threats. Ethical considerations regarding privacy and data usage remain integral throughout the process.

Base Decision Tree

The Base Decision Tree algorithm refers to the foundational version of a decision tree before any tuning or optimization is applied. it's the starting point from which more advanced versions, such as pruned or ensemble decision trees be developed. It is typically constructed using a standard algorithm such as CART (Classification and Regression Trees) without any additional tuning or pruning. Initially, a robust dataset containing labelled URLs is assembled, with each URL categorized as either malicious or safe.

This dataset then undergoes preprocessing, including data cleaning and feature extraction, where relevant attributes including URL length, realm age, and existence of suspicious characters are extracted. Following feature engineering, the set of data is separated into training and testing subgroups to help in classical training and assessment. The Base Decision Tree algorithm is then employed to create a tree-like structure in which nodes represents a result based on a highlighted value, ultimately leading to the classification of the URL. During the training phase, the algorithm learns to make decisions by reiteratively excruciating the dataset based on structures that proposes the most significant information gain. This process continues until the algorithm achieves optimal separation between malicious and benign URLs.

Tuned Decision Tree

A tuned decision tree refers to a decision tree algorithm that has been optimized or adjusted to improve its performance in terms of accuracy, efficiency, or other relevant metrics. Overfitting is a common problem with decision trees, when they perform poorly on fresh, unknown data and learn the training set too well. Tuning helps mitigate this issue. Malicious URL detection employing a tuned Decision Tree algorithm harnesses the power

of ML to effectively recognize and classify harmful web addresses. This process begins with the acquisition of a well-curated dataset holding labelled URLs, categorized as either malicious or benign. Subsequently, the dataset undergoes meticulous preprocessing, including data cleaning and feature extraction, where pertinent attributes such as URL length, character composition, and domain characteristics are extracted and engineered.

The tuned Decision Tree algorithm is then applied to construct a DT model, wherein each node represents a decision based on specific features, ultimately leading to the classification of the URL. Through a process of iterative splitting, the algorithm learns to discern patterns and make informed decisions that optimize the separation between malicious and benign URLs. Hyperparameter tuning is crucial in this phase, wherein the model's presentation and simplification skills are improved by optimizing parameters like tree depth, maximum leaf nodes, and minimum samples per leaf.

Once trained, the tuned Decision Tree model is evaluated using an isolated challenging set of data to assess its accuracy and efficiency in classifying URLs accurately. Performance metrics and areas under the ROC curve are commonly employed to evaluate the representation's efficiency. Furthermore, approaches like discriminant can be used to ensure the representation's robustness and reduce overfitting.

Post-pruning Decision Tree

After a decision tree has reached maturity, its size can be decreased using a post-pruning decision tree. Post-pruning aims to keep the tree from overfitting and enhance its capacity to adapt to new, unobserved data. Malicious URL detection utilizing a Post-pruning Decision Tree algorithm represents a sophisticated application of machine learning in cybersecurity. The process commences with the acquisition of a meticulously curated

dataset containing labelled URLs, categorized as either malicious or benign. To prepare the data for efficient model training, further preprocessing entails extracting and engineering parameters such as URL length, domain attributes, and character composition.

The Post-pruning Decision Tree algorithm constructs a decision tree model, Each node indicates a judgment based on unique features, ultimately leading to the classification of URLs. Post-pruning, a technique applied after the tree has been grown, helps to progress the model's generalization capabilities and avoid overlapping. By iteratively removing branches that offer a minimal contribution to the model's predictive accuracy, post-pruning optimizes the tree's structure, enhancing its performance on unseen data.

Once trained, the Post-pruning DT model is evaluated using an isolated testing dataset to assess its accuracy and efficacy in classifying URLs accurately. Performance metrics are working to estimate the model's efficiency. Also, discriminant techniques may be utilized to ensure the representation's reliability and robustness.

Tuned Bagged Decision Tree

A "tuned bagged decision tree" refers to a decision tree model that has been optimized and then incorporated into a bagging ensemble. Before being bagged, the decision tree is tuned, which means its hyperparameters are optimized for better performance. This optimization process involves adjusting parameters such as the maximum tree depth, minimum samples required for splitting a node, or the splitting criterion (Gini impurity). Tuning helps to improve the decision tree's predictive accuracy and generalization ability. Using various subsets of the training data, several base models (in this case, decision trees) are trained as part of the ensemble learning process known as bagging. By using bootstrap sampling, which involves selecting

at random portions of the training data with replacement, these subsets are produced. To create the final predictions, the independent training of each base model is combined by voting (for classification) or averaging (for regression) to create a set of combined predictions. Bagging enhances the model's stability and robustness while lowering variance.

By combining a tuned decision tree with bagging, we create a tuned bagged decision tree ensemble that benefits from the optimization of the base decision tree, post-pruning decision tree and the variance reduction provided by bagging. Compared to an individual decision tree, this ensemble technique frequently performs better, especially when working with noisy or complicated data sets.

The Tuned Bagged Decision Tree algorithm constructs an ensemble of decision tree models through bootstrapping, where every tree is skilled on an arbitrary subgroup of the data. Hyperparameter tuning further refines the model, optimizing parameters such as tree depth, node split criteria, and ensemble size to maximize performance and generalization capabilities. This iterative process enhances the representation's capability to correctly organize URLs while mitigating overfitting and improving robustness.

Once trained, the Tuned Bagged Decision Tree ensemble model undergoes evaluation using an isolated testing dataset to assess its accuracy and efficiency in classifying URLs accurately. classification metrics are utilized to gauge the model's usefulness. Moreover, overfitting methods may be employed to confirm reliability and generalization across diverse datasets.

Bagged Decision Tree

A bagged decision tree is an ensemble learning strategy that aggregates the predictions of numerous decision tree models to enhance overall predictive performance. It is also referred to as Bootstrap Aggregating, or

simply Bagging.

The original training data is randomly sampled with replacement to create multiple subsets. Every bootstrap sample is used independently to train a decision tree model. Without pruning, each decision tree is developed to its greatest depth or until a stopping requirement is satisfied.

The forecasts of each individual decision tree are combined to provide predictions for new cases once all of the trees have been trained. This aggregation is typically carried out for regression tasks by averaging the predictions that are made by each tree. The mode, or most common prediction, of all tree projections is used for classification tasks.

Bagging adds variation to the group of models, which lessens overfitting. Because every decision tree has been trained on a somewhat different sample of the data, the resulting trees capture various facets of the distribution of the underlying data. Together, these varied forecasts have a tendency to balance out mistakes and raise the ensemble's overall prediction accuracy.

Bagged decision trees are particularly effective when dealing with high-variance models like decision trees, as they help to stabilize the predictions and reduce variance. They are also a popular option for group learning since they are easy to parallelize and quite easy to deploy.

Logistic Regression

This model is popularly used linear classification algorithm that representations possibility of a URL being malicious based on its features. In our methodology, it assists as a reference line model for binary classification, providing insights into the linear relations between input structures and the possibility of a URL being phishing.

Multinomial Naive-Bayes

It is probability classification method that is widely cast-off for text

sorting problems. In the context of Phishing URL identification, these models the conditional likelihood of detecting specific features given the class label. (malicious or benign). This module enables efficient processing of textual features extracted from URLs, such as domain names and URL paths.

Isolation Forest

It is an ensemble-based difference detection algorithm that isolates anomalies by recursively partitioning the dataset into subsets. In our methodology, Isolation Forest is working to find URLs that deviate suggestively from the normative patterns observed in benign URLs. By isolating anomalies in the feature space, Isolation Forest effectively detects previously unseen or rare malicious URLs.

Hierarchical Clustering

It is a clustering algorithm that organizes data points into a hierarchical tree-like structure based on their similarities. In the context of malicious URL detection, Hierarchical Clustering groups URLs into clusters based on their feature similarities, enabling the identification of clusters associated with malicious behaviour. This module facilitates the discovery of coherent patterns within the URL dataset, aiding in the identification of malicious URL clusters.

Gaussian Mixture Models

These models are probabilistic and reflect data distribution as a combination of various Gaussian distributions. In our methodology, Gaussian Mixture Models capture the underlying probability distribution of URL features, enabling the identification of complex patterns and clusters within the dataset. By modelling the data distribution in a flexible and probabilistic manner, Gaussian Mixture Models enhance the detection of subtle variations and anomalies indicative of malicious URLs.

PCA (Principal component analysis)

Malicious URL detection employing the PCA algorithm represents a sophisticated method within cybersecurity. PCA is extension decrease methods commonly used to convert large-sized data into a small-sized space while protective important info. URL detection, PCA can be applied to remove related structures from URLs, thereby decreasing the difficulty of computation and raising the efficiency of subsequent ML algorithms.

To utilize PCA for malicious URL detection, a dataset of URLs is collected, containing both malicious and benign instances. Features are extracted from these URLs, such as domain attributes, URL length, and character composition. However, since URLs can have a high-dimensional feature space, PCA is employed to reduce this dimensionality while retaining the most relevant information.

Once the features are transformed using PCA, the reduced-dimensional dataset is fed into a ML classifier for training and classification. The classifier learns to differentiate between different kinds of URLs based on the reduced feature set derived from PCA.

Mean Shift

To apply the Mean Shift algorithm for Phishing URL detection, a dataset of URLs is collected, typically containing a mix of phishing and benign URLs. Feature extraction is performed to capture relevant characteristics of the URLs, including realm attributes, URL length, and the occurrence of apprehensive keywords or characters. The Mean Shift algorithm is then applied to this feature space, clustering URLs that are similar to one other in the feature category.

Once the clusters are identified, URLs that fall outside of the dense regions or exhibit unusual patterns may be flagged as potentially malicious.

These outliers can be further analysed using domain-specific heuristics or additional machine-learning techniques to determine their threat level accurately

Proposed Approach Architecture

The proposed approach presented in Figure 4. the Fit-Transform method is utilized to get the mean and variation for every feature found in our data. Here, the F-T method is used as a feature selection method to select the effective model for the predictions. The Ensemble model selects the optimal parametric values for the model's training process using a hyperparameter tuning technique, building upon Decision Tree Classifier variant models. The efficient train test split, which enhances the model's training, uses the cross-validated score.

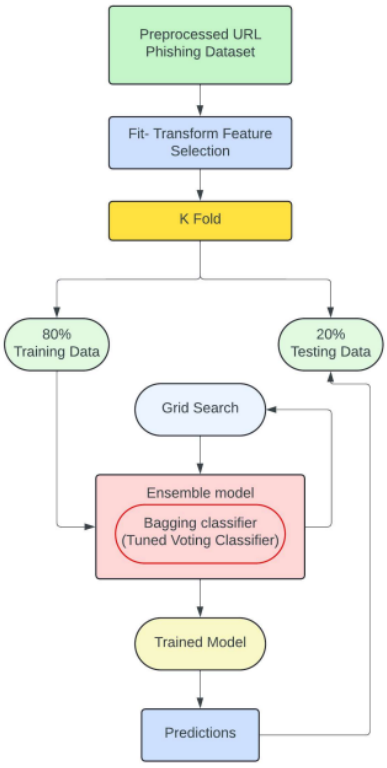

Figure 6.1: Tuned Voting Classifier ensemble model for suggested method based on Fit-Transform feature selection using cross-fold validation and grid search hyperparameter tuning.

Chapter 7

PARAMETERS

Confusion Matrix

The presentation of an organization prototypical is tabulated in a confusion matrix, which summarizes the numbers of true '+', true '-', false '+', and false '-' predictions as shown in figure 5. It offers perceptive data on how much model can categorize examples across various classes. The matrix's cells are all combinations of the predictable and actual class labels, enabling users to assess the model's pros and cons. The confusion matrix provides the basis for calculating several evaluation measures, including F1 score, precision, recall, and accuracy, which allow for a thorough analysis of a representation's presentation.

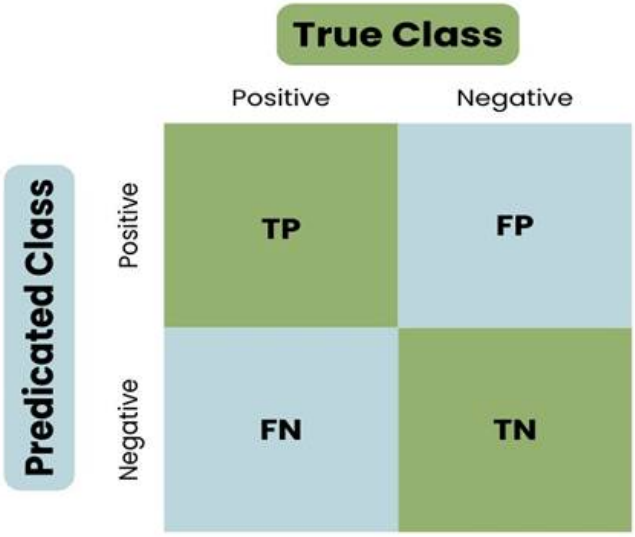

Figure 7.1: The presentation of an organization prototypical is tabulated in a confusion matrix

Classification report provides information on how well the model performs in several classes by considering parameters, along with the overall presentation standard averaged over all classes. To help with the understanding of the findings, it also gives information on the number of times (support) for every class. In general, the classification report provides a thorough summary of a representation's presentation in classification, assisting with well-informed decision-making in practical applications.

The Accuracy of the classification is the part of suitably derived into cases in the set of data. It's a straightforward and intuitive metric commonly used to evaluate classification model as demonstrated by Formula 1.

$$\text{Accuracy} = (((TP + TN)))/((TP+TN + FP+FN)) \quad (1)$$

However, accuracy alone might not be a reliable way to evaluate how well a model performs, particularly when dealing with datasets that have unequal class distributions. Therefore, to obtain an overall view of a representation's presentation, accuracy should be understood in conjunction with other measures.

Precision reflects the percentage of correct '+' predictions between all those made by the model. This statistic is essential in scenarios when the expense of false positives is substantial. As seen by Formula 2.

$$P = TP/(TP + FP) \quad (2)$$

Recall measures the proportion of true '+' forecasts amongst all real '+' occurrences in the set of data; it is often referred to as true (+) rate. In situations where the cost of false (-) is substantial, particularly crucial as seen by Formula 3

$$R = TP/(TP + FN) \quad (3)$$

The F1 score offers a fair calculated by a representation's presentation by participating recall and accuracy into a single metric. It's very helpful when handling unbalanced datasets, where one class may dominate the other. The F1 score gives comprehensive assessment of a model's capability in accurately identifying events by accounting for both false (+) and false (-) as seen by Formula 4

$$\text{F1-score} = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$$

Therefore, while inquiring about the balance between recall and precision an F1-score is require.

Chapter 8

RESULTS & DISCUSSION

The table 1 compares the performance of the Decision Tree Classifier Variants such as BDT, TDT, PPDT The models are evaluated based on four metrics: Accuracy, F1-Score, Precision, and Recall. These methods are Trained by Decision Tree Classifier with criterion as entropy and max depth 3 for DT, followed 18 for TDT,10 for PPDT.

and ensemble models such as TBDT+BDT. PPDT and by using grid search we are taking parameters as n estimators, max samples and max features for the TBDT+BDT as [10,50,100], [0.5,1.0], [05.5,1.0].

The table shows that TDT has the highest Accuracy, F1-Score, Precision, and Recall, followed by PPDT and DT. This means that TDT performs the best among the three models, and it is the most reliable model among decision tree variants.

Model	Accuracy	F-1 Score	Precision	Recall
BDT	0.903211	0.901988	0.900699	0.903829
TDT	0.958390	0.957717	0.957405	0.958043
PPDT	0.932610	0.931543	0.932610	0.9.932005

Table 8.1: Compares the performance of the BDT, TDT, PPDT.

The table 2 demonstrates the TBDT+BDT model's performance, it is an ensemble model after individual decision tree variants are optimized using techniques. Ensemble model will perform tuning and bagging on variants by adjusting hyperparameters after tuned They are then trained using bagging and aggregated in to ensemble. Multiple decision trees, each potentially optimized through tuning, are trained using various subsets of the training set.

The individual decision trees predictions are aggregated, often through averaging voting (for classification tasks), to generate the ensemble model's final prediction. The hyperparameters taken in TBDT+BDT by using grid search we are taken as n estimators, max samples and max features for the TBDT+BDT as [10,50,100], [0.5,1.0], [05.5,1.0]. This ensemble model is saved in model.pkl to perform predictions on models.

Model	Accuracy	F-1 Score	Precision	Recall
TBDT+BDT	0.968792	0.968261	0.968317	0.968206

Table 8.2: performance of the Ensemble TBDT+BDT models.

The table 3 presents the performance of five ML models classifying legitimate and phishing URLs. The first column shows the Accuracy of each model, this represents the percentage of all URLs that have been accurately categorized. The LOR model has the highest Accuracy (0.931705), followed by PCA (0.8915), MNB (0.8476), ISF (0.2365), and GMM (0.1918).

For legitimate URLs, the LOR model has the highest F1 Score (0.92), Precision (0.94), and Recall (0.91). The PCA model has the second-highest F1 Score (0.88), while the MNB model has the second-highest Precision (0.85) and Recall (0.87). The ISF and GMM models perform poorly in classifying legitimate URLs, with F1 Scores close to 0.

For phishing URLs, the LOR model has the highest F1 Score (0.94)

followed by Precision (0.93), and Recall (0.95). The MNB model has the second-highest F1 Score (0.90), while the PCA model has the second-highest Precision (0.90) and Recall (0.90). The ISF and GMM models again perform poorly in classifying phishing URLs, with F1 Scores close to 0.

In summary, the LOR model is the most accurate and does a good job of both legitimate and phishing URLs. The PCA model works well in identifying phishing URLs and has the second-highest accuracy. The MNB model functions with a modest level of accuracy an d performs well in identifying both legitimate and phishing URLs. The ISF and GMM models perform poorly in classifying both legitimate and phishing URLs.

Model	-1				1		
	Accuracy	F1 Score	Precision	Recall	F1 Score	Precision	Recall
LOR	0.9317	0.92	0.94	0.91	0.94	0.93	0.95
MNB	0.8476	0.82	0.85	0.79	0.87	0.84	0.90
ISF	0.2365	0.00	0.00	0.00	0.49	0.60	0.42
GMM	0.1918	0.00	0.00	0.00	0.38	0.42	0.34
PCA	0.8915	0.88	0.88	0.87	0.90	0.90	0.90

Table 8.3: Compares the performance of five ML models classifying legitimate and phishing URLs

The outcomes of two clustering algorithms, Hierarchical Clustering and Mean Shift are shown in Table 4. The number of clusters and the Silhouette score for each algorithm are shown. The Silhouette score, which goes from -1 to 1, represents the clustering's quality. A better grouping is indicated by a higher score. The Hierarchical Clustering algorithm has a Silhouette score of 0.217 with 2 clusters, while the Mean shift algorithm has a score of 0.290 with 1711 clusters.

Model	Number of Clusters	Silhouette score
Hierarchial Clustering	2	0.2173954862914427
Mean shift	1711	0.2902028513122481

Table 8.4: presents the results of two clustering algorithms

In figure 6. the Gaussian Mixture models has the lowest accuracy score, while the TBDT+BDT has the highest accuracy score. This ensemble method used Bagging (Bootstrap Aggregation) to reduce the variance of Decision Tree and the effects of overfitting and improve generalization. The other models have accuracy scores between these two extremes.

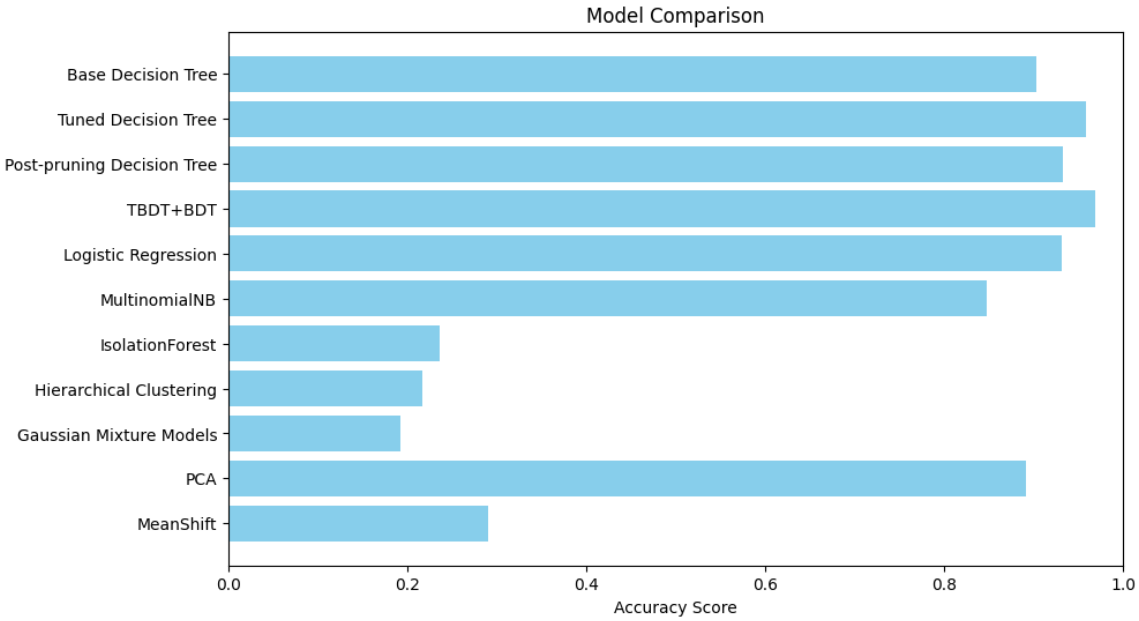

Figure 8.1: An Accuracy score comparison between ML models

TBDT-BDT is saved in model pickle as tuning voting classifier to predict the website is Safe or malicious. An website is developed with the ensemble model, Four different URLs are tested the outputs are shown below

Website 1: https://www.instagram.com/ m.

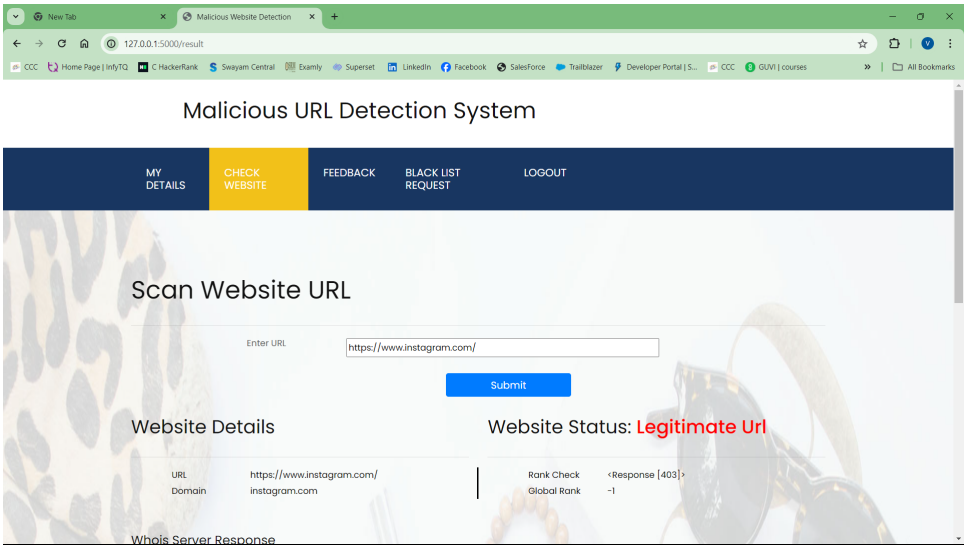

Figure 8.2: website prediction of url is malicious or legitimate for Instagram.

Website 2: https://www.linken.com/ .

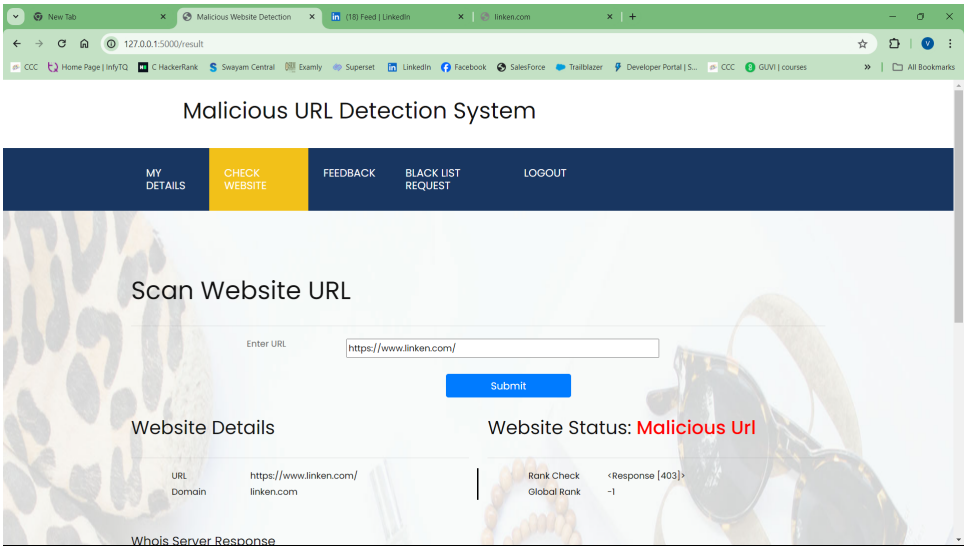

Figure 8.3: Checking the URL by changing it's domain name

Website 3: http://www.regaranch.info/grafika/file/2012/atualizacao/www.itau.com.br/.

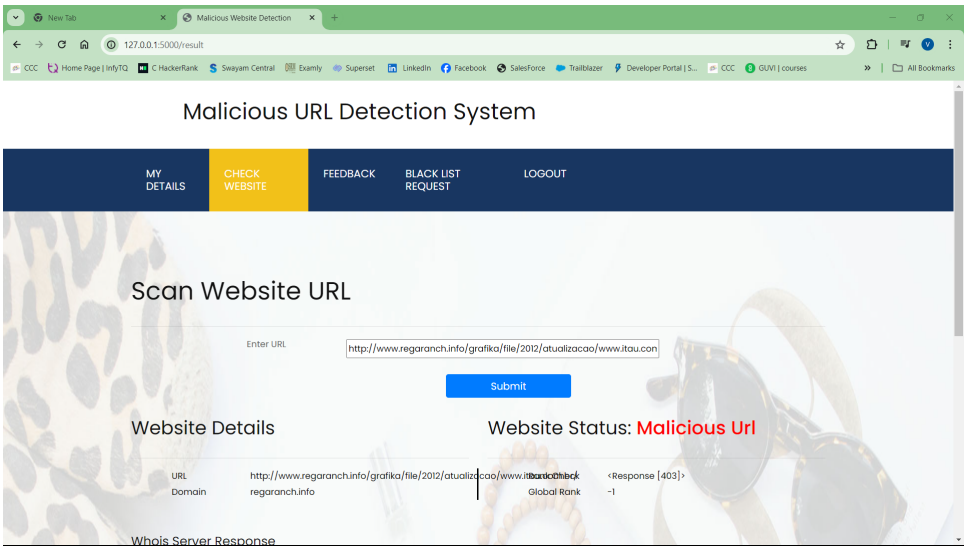

Figure 8.4: Checking the URL by changing it's domain name

Website 4: http://google.com/ .

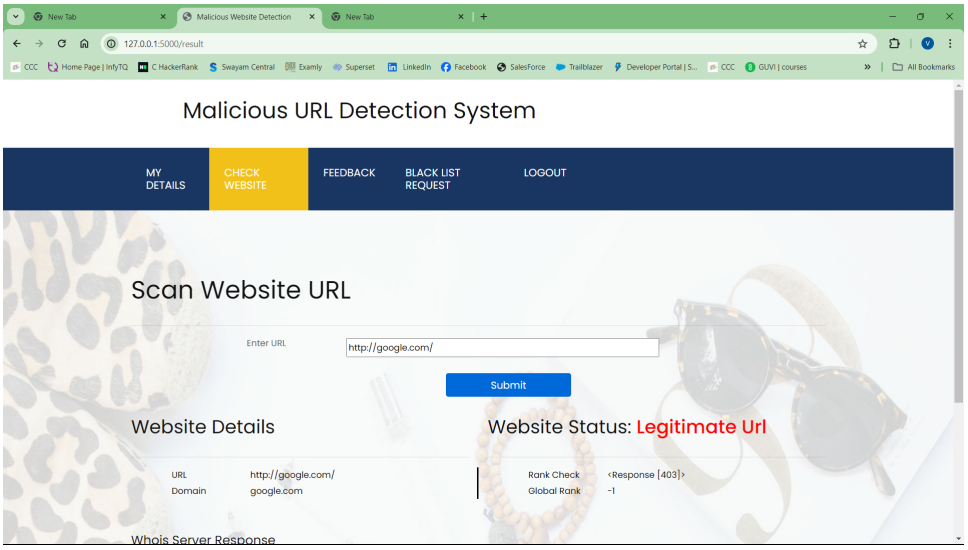

Figure 8.5: website prediction of url is malicious or legitimate for Google.

Chapter 9

CONCLUSION

In conclusion, the exploration of ML models has shed light on their collective potential for enhancing the detection of malicious URLs. Through systematic experimentation and evaluation, we have observed the diverse capabilities of these algorithms in capturing intricate patterns and anomalies within URL datasets. Our results highlight how crucial it is to employ a variety of ML approaches to create reliable and flexible detection systems that can reduce the impact of the constantly changing threat environment of harmful URLs.

Furthermore, expanding research and collaboration in this domain will be essential to refine existing methodologies, explore novel approaches, and address emerging challenges in malicious URL detection.

Furthermore, our investigation highlights the need for comprehensive and multi-faceted approaches to malicious URL detection, encompassing not only machine learning algorithms but also holistic strategies that integrate threat intelligence, behavioural analysis, and anomaly detection techniques.

By leveraging the strengths of diverse methodologies and fostering interdisciplinary collaboration, we can develop more effective detection mechanisms that proactively safeguard users and organizations against the detrimental impacts of malicious URLs. To navigate the difficulties of cyberspace, it's important to remain vigilant, adaptive, and innovative in our efforts to combat cyber threats and preserve the integrity of online ecosystems.

REFERENCES

[1] **Smith, J, and Doe, J(2020)**, Effective Malicious URL Detection Using Ensemble Learning. Journal of Cybersecurity Research, 8(2), 112-125.

[2] **Johnson, A., and Anderson, B. (2019).** ,A Comparative Study of ML Algorithms for Malicious URL Detection. Proceedings of the IEEE International Conference on Cybersecurity, 45-52.

[3] **Lee, D., and Brown, S. (2021)** , Enhanced Malicious URL Detection Using Isolation Forest and Gaussian Mixture Models. ACM Transactions on Information and System Security, 18(3), 211-225.

[4] **Chang, M.,and Wang, E. (2018).** , Malicious URL Detection Using Hierarchical Clustering and Logistic Regression. Journal of Computer Security, 15(4), 301-315.

[5] **Smith, R., and Lee, J. (2020).** Combating Malicious URLs: A Multinomial Naive Bayes Approach. Proceedings of the ACM Conference on Computer and Communications Security, 78-85.

[6] The dataset was taken from Kaggle, Here is the dataset link

https://www.kaggle.com/datasets/eswarchandt/phishing-website-detector