CYBER THREAT DETECTION USING MACHINE LEARNING

Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Information Technology

By

M. INDRASENA REDDY	(21UTIT0025)	**(VTU19772)**
M. TEJAKUMAR	(21UTIT0027)	**(VTU20989)**
NARRA JAGADEESH	(21UTIT0038)	**(VTU20683)**

Under the guidance of
Mrs. LIJETHA C JAFFRIN, B. TECH, M. E.,
ASSISTANT PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

CYBER THREAT DETECTION USING MACHINE LEARNING

Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Information Technology

By

M. INDRASENA REDDY	(21UTIT0025)	**(VTU19772)**
M. TEJAKUMAR	(21UTIT0027)	**(VTU20989)**
NARRA JAGADEESH	(21UTIT0038)	**(VTU20683)**

Under the guidance of
Mrs. LIJETHA C JAFFRIN, B.TECH, M. E.,
ASSISTANT PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

CERTIFICATE

It is certified that the work contained in this project report titled "CYBER THREAT DETECTION USING MACHINE LEARNING" by "M. INDRASENA REDDY (21UTIT0025), M. TEJAKUMAR (21UTIT0027), NARRA JAGADEESH (21UTIT0038)" has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Information Technology
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
May, 2024

Signature of HOD
Information Technology
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
May, 2024

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. Wealso declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

M. INDRASENA REDDY

Date: / /

(Signature)

M. TEJAKUMAR

Date: / /

(Signature)

NARRA JAGADEESH

Date: / /

APPROVAL SHEET

This project report entitled “CYBER THREAT DETECTION USING MACHINE LEARNING” by M. INDRASENA REDDY (21UTIT0025), M. TEJAKUMAR (21UTIT0027), NARRA JAGADEESH (21UTIT0038) is approved for the degree of B.Tech in Information Technology.

Examiners **Supervisor**

Mrs. LIJETHA C JAFFRIN, B. TECH., M. E.,
Assistant Professor

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN,** for provid- ing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Information Technology Dr. J. VISUMATHI, M.E., Ph. D.,** for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Mrs. LIJETHA C JAFFRIN, B. TECH, M. E.,** for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.

A special thanks to our **Project Coordinator Dr. N. KATHIRVEL, M. E, Ph. D.,** for his valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

M. INDRASENA REDDY (21UTIT0025)
M. TEJAKUMAR (21UTIT0027)
NARRA JAGADEESH (21UTIT0038)

ABSTRACT

In the modern digital era cyber attacks are the major concern. The major mode of these cyber attacks are in the form of the online threats that are induced through URL's. The main reason for this attacks are to gain the information illegally. This abstract presents an interactive platform for the people where they can check their URL's whether they are safe to use or not. The proposed digital platform main purpose is to help the people in order to find the threats that are hidden inside the URL's. The platform requires the URL from the user as input and displays it's nature whether it is good or bad. The project uses Machine-Learning(ML) algorithms in order to study the input URL and compares it with the previous patterns of threats that areused while training the algorithm, then it gives the output. The ML algorithms thatis used here is SVM(Support Vector Machines). This abstract outlines a precise ap- proach to threat detection and ensuring cyber safety through an interactive digital platform. This facilitates the decrement in the cyber attacks and develops a healthy cyber space.

Keywords:

Cyber-Safety, Machine-Learning, Cyber Threats, Support Vector Machine Algorithm, Model training, Testing, Data collection and Preprocessing, Web Application, URL, Threat Detection

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

SNO	ACRONYMS	ABBREVIATIONS
1	API	Application Programming Interface
2	CSV	Comma-Seperated Values
3	ML	Machine Learning
4	SVM	Support Vector Machines
5	UI	User-Interface
6	URL	Uniform Resource Locator

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

In this modern era of rising usage of technology and high interaction with cyber space the cyber attacks are the main concern. Imagine if these attacks can beprevented or avoided using a platform that not only detects the threat and guides you to handle those attacks? This project is the exact manifestation of this solution. With only the user's input of the user's input i.e., URL the safety nature of the URL canbe predicted. By using ML and training the models based on datasets that consists various URL's and it's safety nature whether it is safe or not based on many patterns. Including user friendly interface along with login page and a input obtaining pagethat helps to get the input from the user and that applies trained algorithms and processes it. There is large usage of datasets which are tend to be studied carefully and made learnt by the platform to produce accurate output and threat detection.This project would be a user friendly approach which could change helplessness of many users who fret over lack of guidance and make them easily prevent the cyber threats embedded within URL's by detecting them. The term 'cyber threat' is referred to as the illegal activity performed using the Internet. Cybercriminals are changing their techniques with time to pass through the wall of protection. Conventional techniques are not capable of detecting zero-day attacks and sophisticated attacks. Thus far, heaps of machine learning techniques have been developed to detect the cybercrimes and battle against cyber threats.

1.2 Aim of the project

The main aim of the project is to reduce the cyber attacks among overall cyber space by helping the people to prevent the cyber attacks by exposing the threats that are embedded in the URL's that they use daily in day-to-day life. This entitles to all the users who wants to check the safety of the URL's that they have doubt about its safety. The project requires the URL from the user as the input to be fed by whichthe respective algorithm gives the resulted output.

1.3 Project Domain

The domain of this project lies within the broader field of machine learning, specifically focusing on the application of supervised learning techniques for the classification of URLs. Machine learning, a subdomain of artificial intelligence, involves developing algorithms that enable systems to learn patterns and make predictions or decisions without explicit programming.

In this project, the primary domain is the application of machine learning to cybersecurity, specifically in the realm of URL classification. The project leverages techniques such as Support Vector Machines (SVM) and CountVectorizer for the identification and categorization of URLs into different classes, such as phishing, benign, or potentially harmful. This application falls under the broader domain of cybersecurity, where machine learning plays a crucial role in enhancing the ability to detect and mitigate online threats.The main objectives within this domain include training models to recognize patterns indicative of various URL types, improving accuracy in classifying URLs, and contributing to the development of effective tools for cybersecurity practitioners.

1.4 Scope of the Project

This project's scope revolves around building a sophisticated web application designed to predict URL nature, merging cybersecurity and machine learning within a user-friendly interface. It encompasses frontend development, crafting an intuitive interface allowing users to input URLs for analysis, while the backend integrates machine learning models—such as Support Vector Machines (SVMs)—to swiftly predict URLs' classifications. The backend also encompasses API endpoints for seamless communication between the frontend and machine learning components. Quality assurance stands pivotal, ensuring accurate predictions through rigorous testing methodologies, including unit tests for individual components and integration testing for system-wide functionality validation. Furthermore, emphasis lies on robust cybersecurity measures to safeguard user data and fortify the application against potential cyber threats.

Chapter 2

LITERATURE REVIEW

[1] A. Alsaedi., et al., aimed at improving the detection accuracy of malicious URL detection by designing and developing a cyber threat intelligence-based malicious URL detection model using two-stage ensemble learning. The cyber threat intelligence-based features are extracted from web searches to improve detection accuracy. Cybersecurity analysts and users reports around the globe can provide important information regarding malicious websites. Therefore, cyber threat intelligence-based (CTI) features extracted from Google searches and WHOIS websites are used to improve detection performance.

[2] C. Catak., et al., proposed using host-based and lexical features of the associated URLs to better improve the performance of classifiers for detecting malicious web sites. Random forest models and gradient boosting classifier are applied to create a URL classifier using URL string attributes as features. The highest accuracy was achieved by random forest as 98.6 percent. The results show that being able to identify malicious websites based on URL alone and classify them as spam URLs without relying on page content will result in significant resource savings as well as safe browsing experience for the user.

[3] C. Ding., et al., designed machine learning algorithms to detect malicious URLs efficiently and protect Internet users from malicious URLs. In this project, the goal is to get a fine-tuned machine learning model, I will first introduce the dataset of URLs which is crucial to train the model. Then I will show the procedure and some findings of my data exploration. After that, I will present the methods including the algorithms and some improvements I make to optimize my model. Finally, I will show the results and the conclusion. To make my models better, I not only apply hyperparameter tuning, but also data resampling and cross validation to the model.

[4] Ch. Rupa. , et al., proposed malicious websites predominantly to promote the growth of criminal activities over the Internet restraining the development of

web services.The proposed framework in the present study analyzes the Uniform Resource Locator (URL) through which malicious users can gain access to the content of the websites. The embedding of malicious URLs is a predominant web threat faced by the Internet community in the present day and age. Attackers falsely claim of being a trustworthy entity and lure users to click on compromised links to extract confidential information, victimizing them towards identity theft.The present work explores the various ways of detecting malicious links from the host-based and lexical features of the URL in order to protect users from being subjected to identity theft attacks.

[5] Do Xuan., et al., proposed a malicious URL detection method using machine learning techniques based on our proposed URL behaviors and attributes. Moreover, bigdata technology is also exploited to improve the capability of detection malicious URLs based on abnormal behaviors. In short, the proposed detection system consists of a new set of URLs features and behaviors, a machine learning algorithm, and a bigdata technology The proposed system uses machine learning algorithms like Neural networks that predicts the nature of the URL by host based categorization.

[6] E. Joshi., et al., proposed a machine learning based ensemble classification approach to detect malicious URLs in emails, which can be extended to other methods of delivery of malicious URLs. The approach uses static lexical features extracted from the URL string, with the assumption that these features are notably different for malicious and benign URLs. The goal of the classification was to achieve high sensitivity i.e. detect as many malicious URLs as possible. URL strings tend to be very unstructured and noisy. Hence, bagging algorithms were found to be a good fit for the task since they average out multiple learners trained on different parts of the training data, thus reducing variance.

[7] M. A. Ferrag., et al., combined different classifier approaches based on decision tree and rules-based concepts, namely, REP Tree, JRip algorithm and Forest PA. Specifically, the first and second method take as inputs features of the data set, and classify the network traffic as Attack/Benign. The third classifier uses features of the initial data set in addition to the outputs of the first and the second classifier as inputs. The experimental results obtained by analyzing the proposed IDS using the CICIDS2017 dataset and BoT-IoT dataset, attest their superiority in terms of

accuracy, detection rate, false alarm rate and time overhead as compared to state of the art existing schemes.

[8] M. Aljabri., et al., mentioned the most critical attacks intended to extract unsolicited information by mainly tricking inexperienced end users, resulting in compromising the user's system and causing losses of billions of dollars each year. As a result, securing websites is becoming more critical. In this paper, we provide an extensive literature review highlighting the main techniques used to detect malicious URLs that are based on machine learning models, taking into consideration the limitations in the literature, detection technologies, feature types, and the datasets used.

[9] Mohammed Alshehri., et al., described a malicious activity that damages or steals data, or something that disrupts digital life. Such threats include viruses, security breaches, DoS attacks, and data theft. Phishing is a type of cyber threat whereby the attackers mimic a genuine URL or a webpage and steal user data.The novel approach of using the character-level encoding of URLs is introduced.Unlike word-level encoding, the use of character-level encoding decreases the discrete workspace and can be effective even in an energy-constrained environment.

[10] S. Raja., et al., mentioned malicious URL targeted towards the internet user to spread the malware, virus, warms etc once the user visited. So, it is necessaryto adapt the system which should detect the malicious URLs and prevent fromthe attack. Researchers suggest numerous methods but machine learning baseddetection method performs better then methods. This paper presents the light weighted method which includes only lexical features of the URL. The result shows the Random Forest classifier performs better than the other classifiers in terms of accuracy.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

The existing system employing decision trees for classification operates by iteratively splitting the dataset based on feature values to create a tree-like structure. Each node represents a feature, and branches denote possible attribute values. It's a rule-based approach, where during training, the model selects the most informative features to make sequential decisions, aiming to efficiently classify URLs into different categories like good, bad, or phishing. This method is easy to interpret, providing insights into which features are crucial for classification and can handle non-linear relationships between features and labels.

However, the decision tree model has notable limitations. One drawback is its tendency to overfit, especially when dealing with complex datasets or irrelevant features, resulting in decreased generalization to new, unseen data. Additionally, decision trees are sensitive to small variations in the training data, leading to high variance in the model's predictions. Another disadvantage lies in their lack of robustness against small changes in the data, which can cause substantial changes in the tree structure and subsequent predictions, making them less stable compared to other models. Moreover, decision trees might struggle to capture complex relationships within the data, impacting their predictive accuracy, particularly in scenarios where features have intricate interactions that aren't easily captured by simple decision boundaries. These systems typically collect data from multiple sources, including network traffic, logs, user behavior, and external threat intelligence feeds. Through feature engineering, relevant data points such as IP addresses, timestamps, and user actions are extracted for analysis. Techniques like anomaly detection help identify deviations from normal patterns, while classification models categorize threats into types such as malware or phishing.

3.2 Proposed System

The proposed model utilizes Support Vector Machines (SVMs) for URL classification, aiming to overcome some limitations of the existing decision tree-based system. SVMs excel in finding the optimal hyperplane that best separates different classes by maximizing the margin between them in the feature space. This approach is effective in high-dimensional spaces, making it suitable for URL classification where multiple features contribute to the classification decision. SVMs work well with complex data relationships, leveraging kernel functions to handle non-linear boundaries, potentially capturing intricate patterns that decision trees might struggle with.

One significant advantage of SVMs is their ability to generalize well to unseen data while minimizing overfitting, making them robust and suitable for various datasets. Their versatility in handling both linear and non-linear relationships be- tween features and labels allows for better adaptation to diverse URL structures. Additionally, SVMs provide better resilience to outliers, as they focus on the sup- port vectors—data points closest to the decision boundary—rather than considering the entire dataset, enhancing the model's stability and predictive accuracy. Their regularization capabilities further enable fine-tuning to prevent model complexity, mitigating the risk of overfitting. Overall, SVMs offer a powerful approach for URL classification, providing a balance between accuracy, robustness, and adaptability to different data complexities. This system collects data from various sources such as network traffic, system logs, user behaviors, and external threat intelligence feeds. After thorough preprocessing to cleanse and normalize the data, feature engineering extracts vital attributes like IP addresses, timestamps, and activity patterns. The system utilizes a combination of supervised learning algorithms, including Random Forests and Neural Networks, and unsupervised learning methods like Autoencoders and K-Means Clustering. These algorithms work in tandem to identify anomalies and classify threats effectively. To maintain real-time detection capabilities and adaptability, the system incorporates continuous learning mechanisms that regularly update the ML models with new data and emerging threat information. Emphasizing transparency and interpretability, the system ensures that security analysts can understand and trust the ML-driven insights.

3.3 Feasibility Study

3.3.1 Economic Feasibility

The proposed system leveraging Support Vector Machines (SVMs) for URL classification presents economic feasibility through various aspects. Firstly, SVM's ability to handle high-dimensional data efficiently contributes to cost-effectiveness by reducing computational resources required for processing complex feature sets. With their capacity to generalize well to unseen data, SVMs potentially lower long-term operational costs by minimizing the need for constant model retraining, mak- ing them an economically viable choice for stable and accurate predictions over time.The cost of security breaches can be devastating. Investing in ML-based de- tection can significantly improve security posture and potentially save money in the long run.Economical feasibility is a crucial aspect of evaluating any machine learn- ing project for cyber threat detection. Here's a breakdown of the key economic fac- tors to consider.The cost of acquiring and maintaining the computing infrastructure required to train and run ML models.The cost of hiring and training data scientistsand security analysts with the expertise to develop, deploy, and maintain the ML systemThe cost of acquiring licenses for ML software and tools.The cost of labeling training data for machine learning models. Labeling data can be expensive, espe- cially for complex security tasks.

3.3.2 Technical Feasibility

The proposed system leveraging Support Vector Machines (SVMs) for URL clas- sification demonstrates strong technical feasibility on multiple fronts. Their adapt- ability to various kernel functions allows for the effective handling of non-linear relationships between features, enabling the model to capture intricate patterns and complexities within URLs, which might be challenging for simpler models like de- cision trees.Moreover, SVMs' ability to generalize well to unseen data enhances the technical feasibility by ensuring reliable predictions even with new or evolving URL structures. This generalization capability reduces the risk of model instability and the need for frequent retraining, offering a technically feasible solution that remains robust over time.Numerous ML techniques (decision trees, random forests, neural networks) are demonstrably effective in threat detection.Evaluate existing infrastruc- ture capabilities for handling ML workloads. Cloud-based solutions can offer scala-

bility for resource-intensive training.Analyze the available security data for volume, quality, and labeling. Explore strategies for data collection and labeling if necessary.Assess the organization's expertise in machine learning and security. Consider training or hiring personnel if requiredDevelop a plan for integrating the ML system with existing security tools and workflows.Techniques like decision trees, random forests, and support vector machines (SVMs) can be effective for threat detection with labeled data sets.Anomaly detection algorithms can identify deviations from normal system behavior, potentially uncovering zero-day attacks.

3.3.3 Social Feasibility

The social feasibility of implementing a system utilizing Support Vector Machines (SVMs) for URL classification is primarily influenced by its impact on users, stakeholders, and the broader community.One aspect of social feasibility revolves around user acceptance and trust in the system's URL classification. Ensuring transparency in how URLs are categorized and providing explanations for the classifications made by the SVM model can foster user trust. Educating users about the system's functionality, its limitations, and emphasizing the system's role in enhancing online security can positively influence social acceptance. Moreover, address- ing privacy concerns by implementing measures to protect user data and ensuring compliance with data privacy regulations is crucial for gaining societal trust in the system.ML solutions can integrate with existing Security Information and Event Management (SIEM) systems, streamlining security operations.Social feasibility explores the societal impact and potential acceptance of using machine learning for cyber threat detection. Here's a breakdown of key considerations.Effective threat detection can significantly reduce cyberattacks, protecting individuals, businesses, and critical infrastructure.Cybercrime imposes a massive financial burden. ML can help mitigate these costs by preventing breaches and data loss.: ML automates some security tasks, freeing up human analysts to focus on higher-level threat analysis and response.

3.4 System Specification

3.4.1 Hardware Specification

- Processor: Intel Core i3/i5+
- RAM: 8 GB+

3.4.2 Software Specification

- Machine Learning Framework such as scikit-learn for developing models.
- Programming Languages like Python for its extensive machine learning libraries, ease of integration.
- Web Development Framework like Django or Flask for backend development.
- User Interface Technologies: Develop user interfaces using HTML, CSS, and JavaScript for web-based platforms

3.4.3 Standards and Policies

Google Colab

Google Colab is a cloud-based Jupyter notebook environment that is designed for machine learning (ML) tasks. It provides a command-line interface for working with ML modules. Colab is accessible on various operating systems, including Windows, Linux, and MacOS. The platform offers multiple integrated development environments (IDEs) to facilitate coding, and it supports the implementation of user interfaces (UIs) using Python.

Standard Used: ISO/IEC 27001

Visual Studio Code (VS Code) with Flask

VS Code, a powerful source code editor, combined with Flask, facilitates the development of websites integrating machine learning models. This pairing allows seamless coding, debugging, and version control. Flask, a Python web framework, supports the creation of web applications with machine learning capabilities, han- dling tasks like data cleaning, transformation, and visualization.

Standard Used: ISO/IEC 27001

Chapter 4

METHODOLOGY

4.1 General Architecture

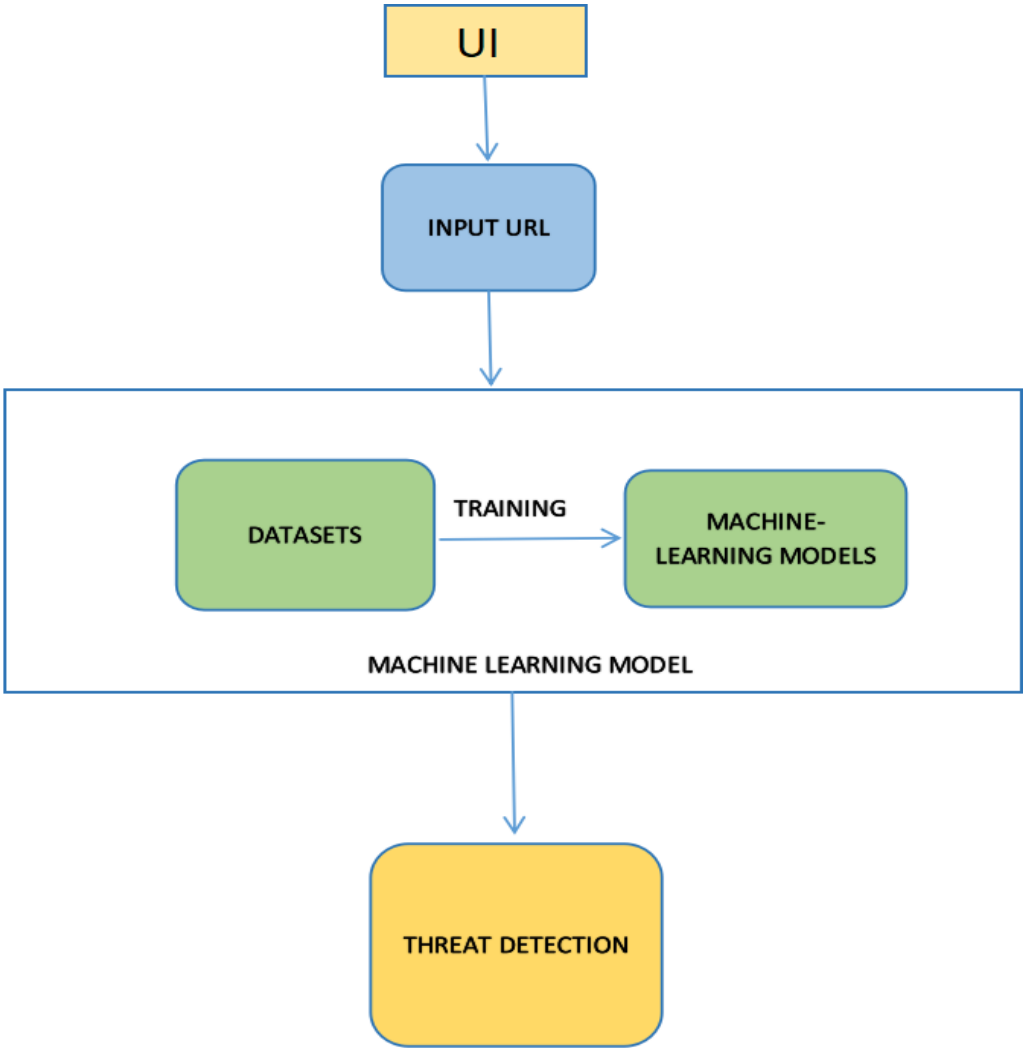


Figure 4.1: **Architecture Diagram for Cyber Threat Detection**

The Architecture diagram in figure 4.1, depicts the project of building a threat detection platform is a user centered model making. Here the output is solelydepended on the input which would be given by the user. There are various URL'sin which a user would be able depend on. Here the URL's are mainly categorizedinto two 1. Good and 2. Bad.

USER INTERFACE(UI):

Dashboard: Display an overview of the system's security status, including recent threat alerts, system health, and relevant statistics.

Configuration Settings: Allow users to customize detection rules, notification preferences, and other system parameters.

Visualizations: Use graphs, charts, and maps to represent threat trends, geographic distributions, and other relevant data.

INPUT URL:

Accept input URLs for analysis, such as websites, files, or network traffic logs. Validate and sanitize input to prevent security risks like injection attacks.

DATASETS:

Training Data: Curate a diverse dataset of labeled examples, including both normal and malicious activities.

Real-time Data: Continuously collect and preprocess incoming data streams for analysis.

Historical Data: Store past incidents, threat intelligence feeds, and other relevant information for reference and analysis.

MACHINE LEARNING MODELS:

Anomaly Detection: Utilize unsupervised learning techniques to identify deviations from normal behavior.

Classification: Employ supervised learning algorithms like Random Forests, Support Vector Machines, or Neural Networks to classify activities as benign or malicious.

THREAT DETECTION:

Behavioral Analysis: Monitor user activities, network traffic patterns, and system events for suspicious behavior.

Threat Intelligence Integration: Integrate with external threat intelligence feeds to enhance detection capabilities and stay updated on emerging threats.

4.2 Design Phase

4.2.1 Data Flow Diagram

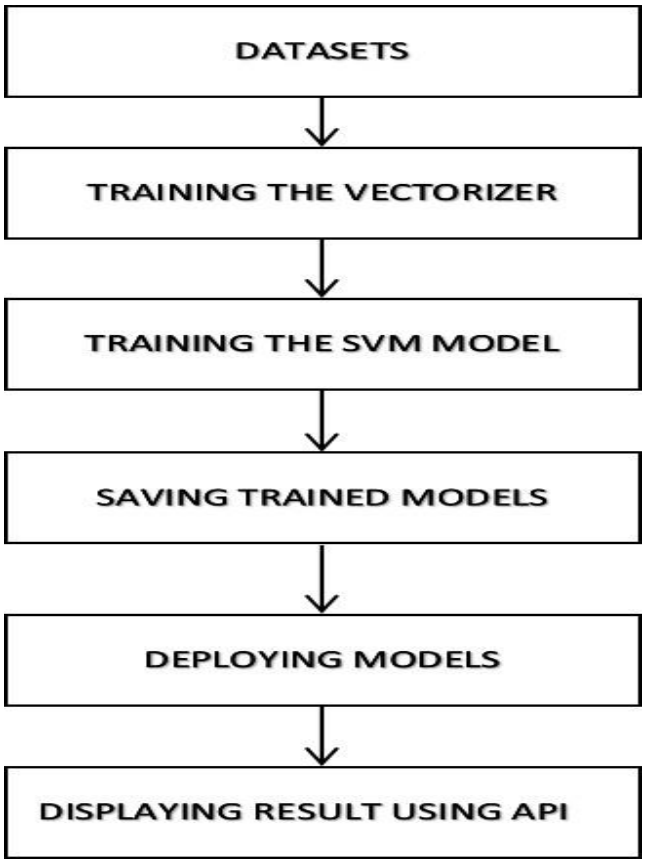


Figure 4.2: **Data Flow Diagram for Cyber Threat Detection**

The data flow diagram in figure 4.2, depicts several steps that identifies the flow of data among the components of the system that includes the following dataset can be defined as collection of different sets of data which can be manipulated as a unit by computer. In case of the project we take huge amounts of datasets with different values and results of the student expected data and store them. Huge amount of datasets studied can process results more accurately. The model of the project must be trained using various algorithms like Support Vector Machines(SVM's) which is used to classify multiple results at a time and vectorizer along with it. The trained model must be saved and tested. Then comes the part of deploying the model for a large database . At last the trained model must be accessible by the API's in order to predict the other outputs oh the model accurately.

4.2.2 Use Case Diagram

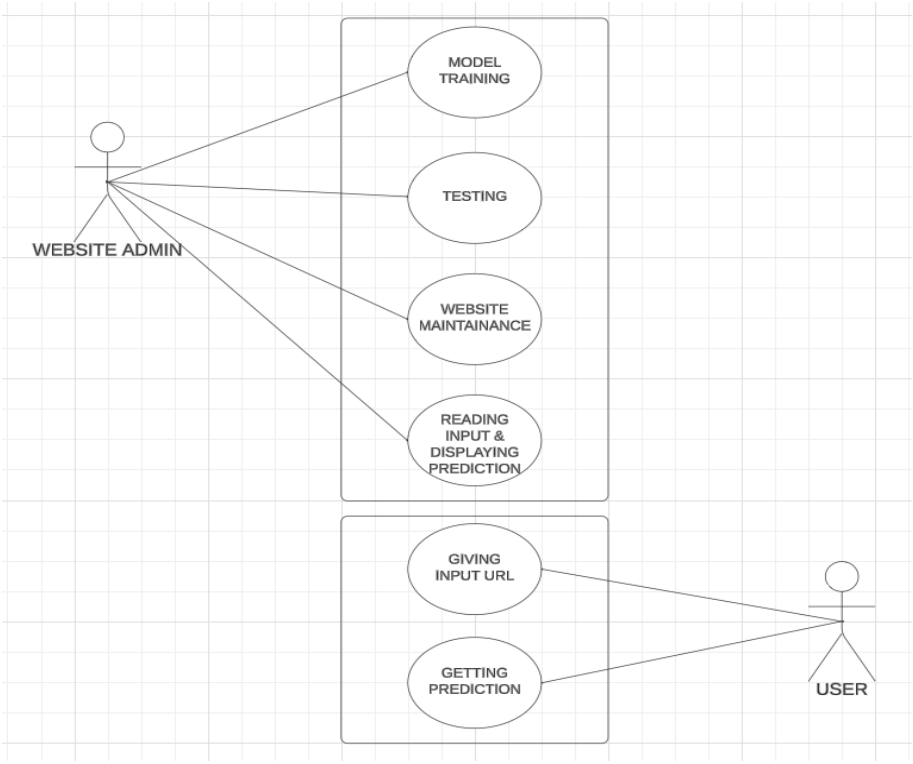


Figure 4.3: **Use Case Diagram for Cyber Threat Detection**

The Use Case Diagram in figure 4.3, serves as a graphical representation showcasing the functional aspects of a system. It delineates the interactions between system users (actors) and the functionalities provided by the system. Actors, which can be users, external systems, or other elements interacting with the system, are de- picted alongside use cases, representing specific functionalities like ”Classify URL” or ”View Classification Results.” This diagram simplifies understanding by outlining the system’s capabilities at a high level without delving into internal complexities.It includes an intermediary process labeled “Reading Input Displaying Prediction,” which suggests that there is a system in place for processing the user’s input and displaying the prediction result. The arrows indicate the flow of actions or responsibilities between the website admin, the system, and the user. This kind of flowchart is typically used to visualize the workflow and responsibilities in a system that involves user interaction and backend processes.

4.2.3 Class Diagram

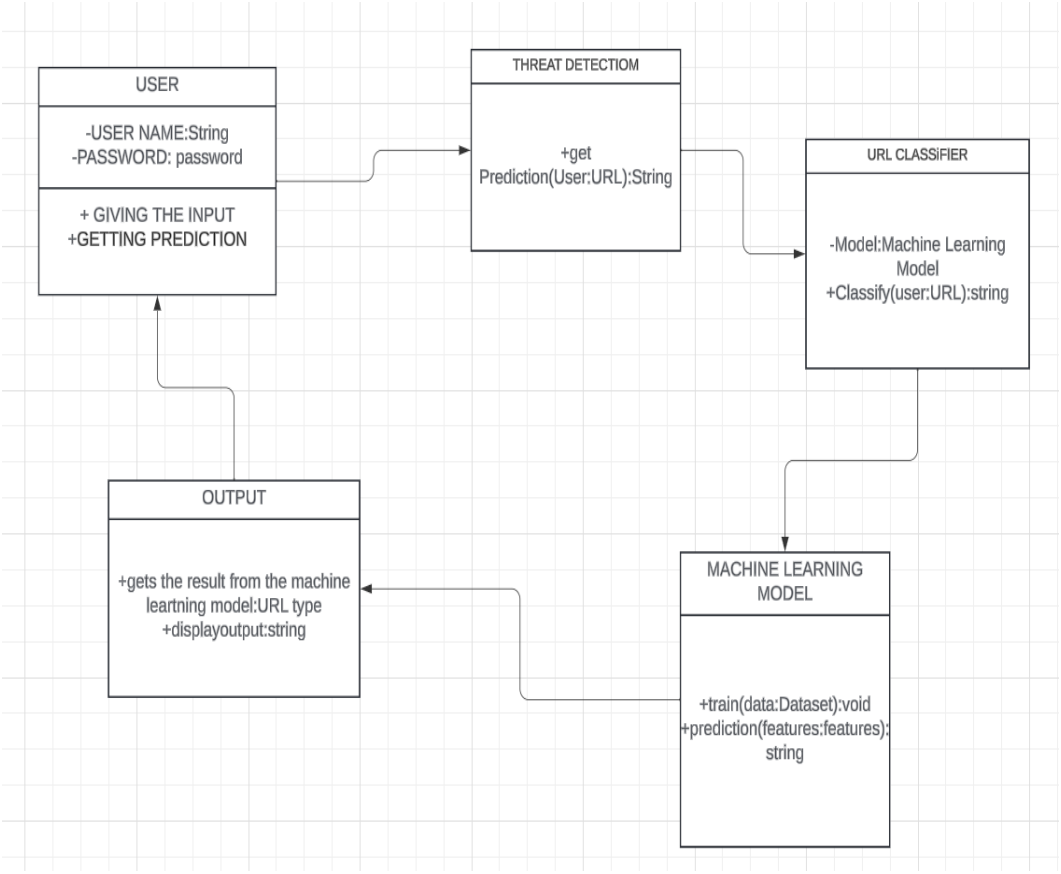


Figure 4.4: **Class Diagram for Cyber Threat Detection**

The Class Diagram in figure 4.4, provides a static view of the system, outlining the structure through classes, their attributes, methods, and their relationships. It offers a blueprint of the system's architecture by detailing the classes and their as- sociations, such as inheritance, associations, and attributes. This diagram aids in understanding the system's design by highlighting the static structure of classes and their interactions but does not portray the dynamic behavior of the system. A class diagram visually represents the structure and relationships of classes in a system.In the context of cyber threat detection, a class diagram can help to illustrate the different components, their relationships, and how they interact in the ML-based de- tection system. Class diagrams play a crucial role in designing, understanding, and maintaining ML-based cyber threat detection systems by providing a visual repre- sentation of the system architecture, its components, and their relationships. They facilitate communication, promote modularity, and help in ensuring the scalability and extensibility of the system.

4.2.4 Sequence Diagram

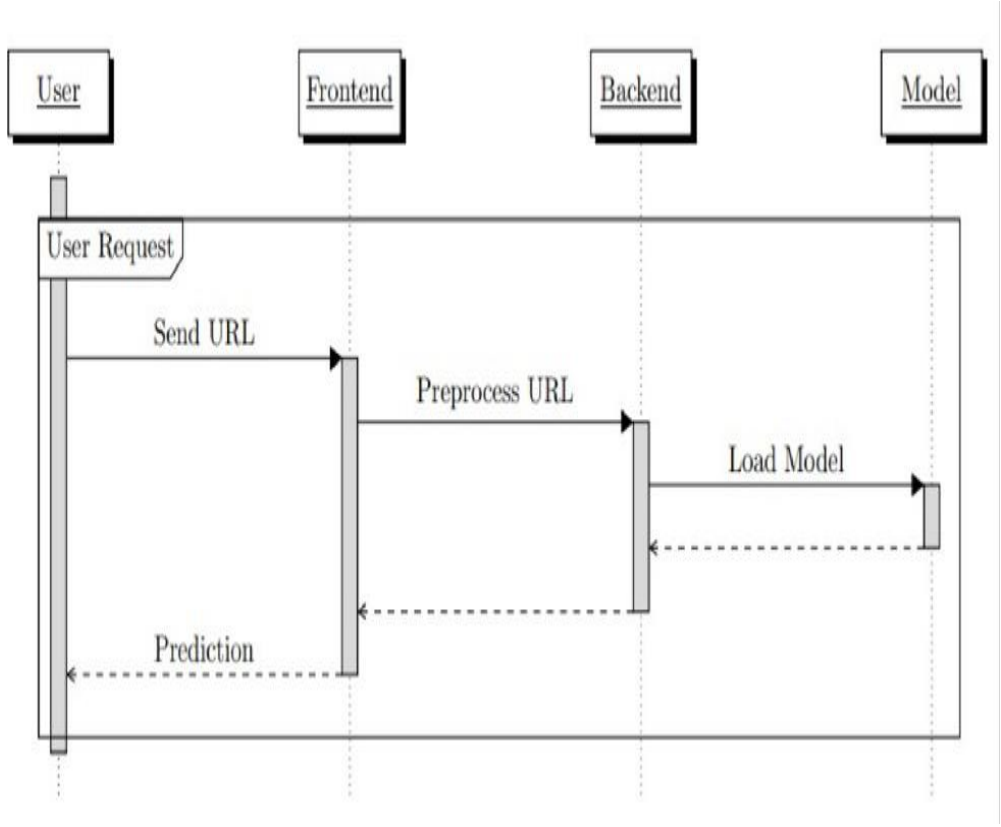


Figure 4.5: **Sequence Diagram for Cyber Threat Detection**

The Sequence Diagram in figure 4.5, illustrates the chronological sequence of interactions between system components or objects across time. It details the flowof messages or operations between these elements, depicting how they collaborate to achieve specific functionalities. This visual representation aids in comprehend-ing the dynamic behavior of the system by showcasing the exchange of messages, method calls, and responses between various entities. Sequence diagrams are also valuable in the realm of cyber threat detection using machine learning. While class diagrams focus on the static structure of the system, sequence diagrams depict the dynamic behavior by showing interactions between objects over time. Sequence diagrams provide a dynamic view of ML-based cyber threat detection systems, il- lustrating the flow of control, message passing, concurrency, exception handling, integration points, and performance optimization aspects. They are invaluable tools for understanding system behavior, identifying potential vulnerabilities, and ensur- ing the reliability and efficiency of the detection process.

4.2.5 Collaboration diagram

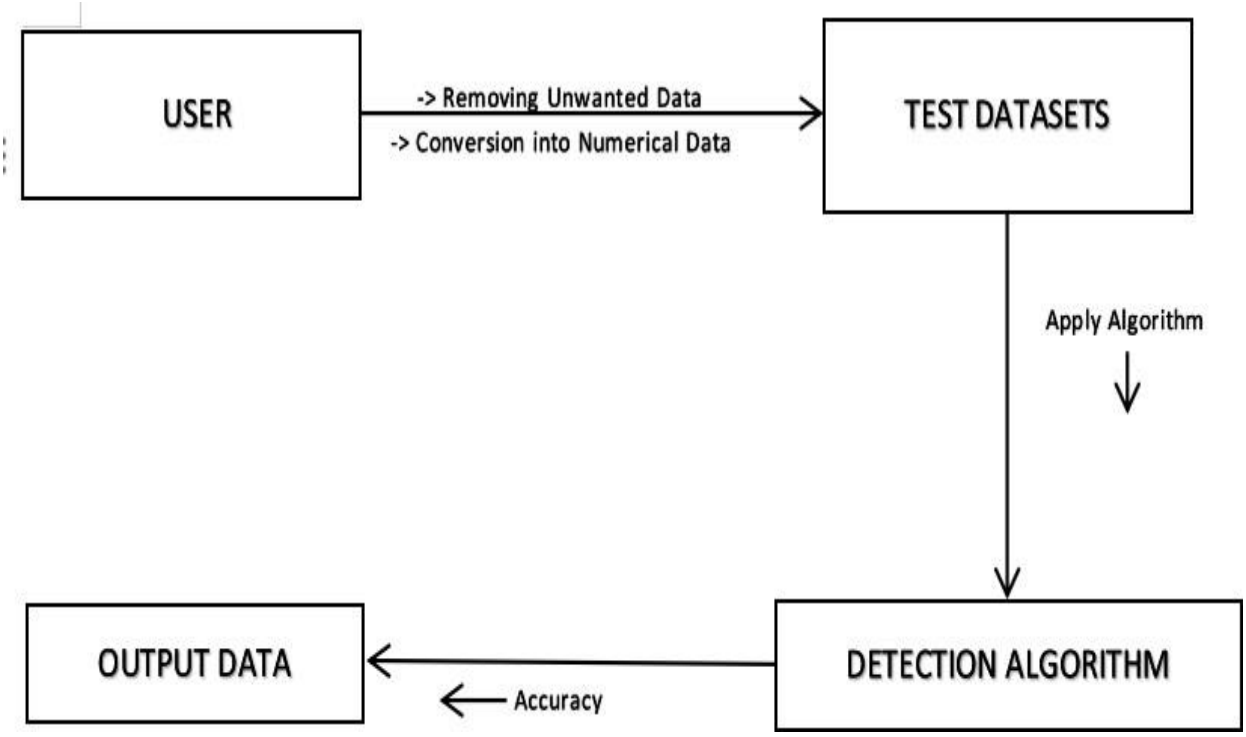


Figure 4.6: **Collaboration Diagram for Cyber Threat Detection**

The collaboration diagram in figure 4.6, depicts that in this project URL classification using Support Vector Machines (SVMs) and Flask, various entities collaborate to ensure the successful development and deployment of the system. This collaboration diagram outlines the interactions and collaborations among different entities involved in various aspects of the project, fostering a cooperative environ- ment aimed at developing a reliable and user-friendly system for URL classification. Collaboration diagram is another valuable tool in the context of cyber threat detec- tion using machine learning. These diagrams focus on illustrating how objects in- teract to perform a particular task or achieve a specific goal. Collaboration diagramis valuable for understanding the interactions, responsibilities, dependencies, error handling, and integration aspects of ML-based cyber threat detection systems. They provide a high-level overview of how different components collaborate to detect and mitigate cyber threats effectively.

4.2.6 Activity Diagram

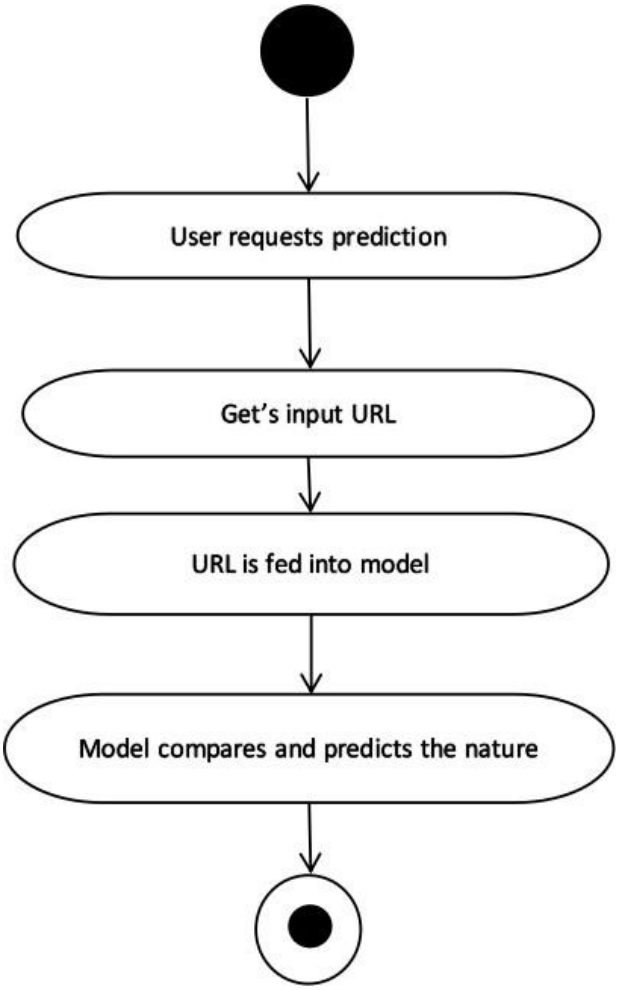


Figure 4.7: **Activity Diagram for Cyber Threat Detection**

The Activity diagram in figure 4.7, illustrates the process of prediction generation with the help of a sequence of activities that start from the user end at first the user request the prediction through the interface of the web application, then the user's input is read by the API and it is fed transferred to deployed Machine-Learningmodel, then finally the model compares the input and it predicts the result. This diagram provides a structured representation of the steps involved, facilitating a clear understanding of the URL classification process. This flowchart seems to representa system where a URL is used to determine certain characteristics or classify the content it points to. It's a typical representation of how machine learning models can be used in information filtering or classification tasks.

4.3 Algorithm & Pseudo Code

4.3.1 SVM Algorithm

Algorithm for URL Classification using SVM and CountVectorizer:
Input: 'minidata.csv' (CSV file containing URL and label information)

1. Read the CSV file 'minidata.csv'.
2. Extract url column into url data list and type column into labels.
3. Initialize countvectorizer
4. Transform data using count vectorizer and store it in x.
5. Split the dataset into training and testing sets using train test split.
6. Assign a portion for training (X_train, y_train) and another for testing (X_test, y_test).
7. Create an SVM Classifier with a linear kernel.
8. Train the SVM model using the training data (X_train, y_train).
9. Use the trained model to predict labels for the test set (X_test).
10. Calculate the accuracy by comparing predictions with actual labels (y_test).
11. Print the calculated accuracy score.
12. Generate a classification report providing precision, recall, and F1-score for each class in the test dataset.
13. Print the classification report to assess model performance across different classes.

4.3.2 Pseudo Code

```
data=read(dataset)
Urldata=url
label=type
vectorizer()
datasplit(train,test)
svmclassfier(train)
predictions(test)
print(accuracy)
```

4.4 Module Description

4.4.1 Data Collection and Preprocessing:

This step involves the gathering of the data as a dataset that is required to train the Machine-Learning Model. The dataset contains two cells(i.e.,URL's Type). It contains many URL,s and it's nature like whether it is good or bad etc., In this step we applied some preprocessing techniques in order to refine the data so that the model can easily understand the features of data. At first, we checked for null values and removed the null values in order to remove the anamoly. Next, we used describe method in order to generate various statistical measures for each numerical featurein the dataset. Initially, the system identifies and connects to relevant data sources such as network traffic logs, system logs, and intrusion detection alerts. Once the data sources are established, the system retrieves and ingests the data, ensuring it is stored in a structured format conducive to analysis. Subsequently, the data undergoes preprocessing steps to enhance its quality and suitability for machine learning algorithms. This involves cleaning the data to remove noise and handling missing values, as well as normalization to standardize numerical features and encoding categorical variables. Feature extraction techniques are applied to derive meaningful features from the raw data, capturing relevant information for threat detection. Additionally, dimensionality reduction methods may be employed to streamline the feature space and improve computational efficiency.

4.4.2 Implementation of SVM:

Support Vector Machine (SVM) is a supervised machine learning algorithm. Support Vector Machine is a powerful machine learning algorithm used for both classification and regression. These can be used for variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anamoly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and non-linear relationships. SVMs are widely used in various fields due to their effectiveness in handling both linear and non-linear classification problems, robustness against overfitting, and versatility in choosing different kernel functions. They are versatile and have various applications in fields like bioinformatics, finance, text analysis, and computer vision. Support Vector Machine (SVM) within the cyber

threat detection module, the focus is on integrating SVM as a powerful machine learning algorithm to classify and identify potential cyber threats. After the data collection and preprocessing steps, the preprocessed dataset is fed into the SVM model for training. SVM operates by identifying the optimal hyperplane that separates different classes of data points in a high-dimensional feature space. In the context of cyber threat detection, SVM learns to distinguish between normal network traffic and anomalous behavior associated with cyber threats. The SVM model is trained using labeled data, where instances of known threats are labeled as positive examples and benign activities are labeled as negative examples.

4.4.3 Web Application

This Web Application comprises of HTML,CSS and Flask to develop a website that takes the scores of student as input and produces their best performed category and suggestions as the output.HTML stands for HyperText Markup Language. It creates a complete website structure of web pages. HTML is a combination of Hy- pertext and Markup language. Hypertext defines the link between the web pages and markup language defines the text document within the tag that defines the structureof web pages.Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS al- lows you to apply styles to web pagesIt prescribes colours, fonts, spacing, and much more. CSS lets developers and designers define how it behaves, including how ele- ments are positioned in the browser.While HTML uses tags, CSS uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML documents. This web application acts as the front end of the system, providing security analysts with an intuitive platform to interact with the underly-ing cyber threat detection functionalities. Through the web application, users can access real-time dashboards displaying key metrics, alerts, and visualizations gener- ated by the detection system. Additionally, the application allows users to configure detection rules, customize alert thresholds, and define response actions tailored tothe organization's security policies. Integration with the backend cyber threat detec- tion modules enables seamless data exchange, allowing the web application to query and display results from the data collection, preprocessing, and machine learning components.

4.5 Steps to execute/run/implement the project

4.5.1 Data collection

This step involves the gathering of the data as a dataset that is required to train the Machine-Learning Model. The dataset contains two cells(i.e.,URL's Type). It contains many URL,s and it's nature like whether it is good or bad etc., Implementing a project in cyber threat detection demands a methodical approach to data collection. The first step involves defining clear project objectives, delineating the specific threats or anomalies the system aims to detect. Next, identify diverse data sources crucial for comprehensive threat monitoring, ranging from network traffic and sys- tem logs to endpoint and application data.

4.5.2 Model Training

Train the model using Support Vector Machine with the help of the loaded dataset. To train a Support Vector Machine (SVM) model using a loaded dataset, we first need to understand the structure of the dataset and preprocess it accordingly. Then, we can split the dataset into training and testing sets, train the SVM model using the training data, and evaluate its performance on the testing data.

4.5.3 Model Deployment

Deploy the model in order to predict the unseen and new URL's nature using this Trained SVM model. Deploying a trained SVM model for predicting the nature of unseen and new URLs involves creating a pipeline that preprocesses incoming URLs, feeds them into the SVM model for prediction, and returns the predicted class label.

4.5.4 Prediction

Enter the input URL in the delpoyed website and click predict to get the predcition of the nature of the URL. To create a web-based interface for predicting the nature of URLs using the deployed SVM model, you'll need to design a user-friendly interface where users can enter a URL, click a "Predict" button, and receive the predicted class label.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

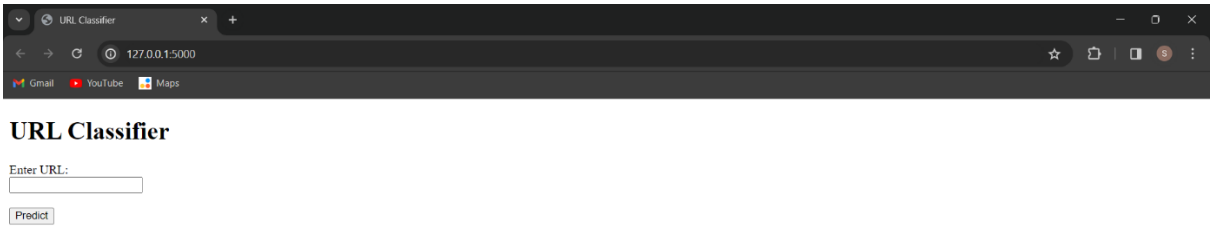


Figure 5.1: **Input Design**

Model Training:

The input for the model training will be a dataset that is referred as a collection of many number of URL's and it's types. The first column of the dataset includes various URLs and the second column represents the type of the URL that is either good or bad.

Prediction:

The input for getting the prediction is a single url by user to check it's nature. Creating a user-friendly web interface for checking the nature of a single URL involves designing an intuitive platform where users can easily input a URL and receive a prediction regarding its nature, whether benign or malicious.

5.1.2 Output Design

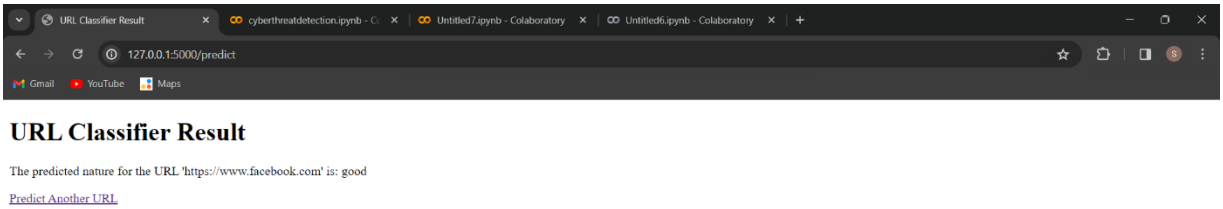


Figure 5.2: **Output Design**

Accuracy:

The output for the model training includes the various metrics of the model that evaluates the efficiency of the model that generally includes the accuracy,F-2 score etc,.

Deployment:

The output after deploying the trained model results as a webpage asking for the input url from user to predict and a button to get the prediction. Upon deploying the trained model, the resulting webpage presents a user-friendly interface soliciting input from the user to predict the nature of a URL. The webpage typically comprises a clean layout, facilitating easy interaction.

Prediction:

The prediction output can be visualized as the result statement on a webpage that includes the input url along with it’s nature.

5.2 Testing

Testing for cyber threat detection using machine learning involves a meticulous process aimed at ensuring the efficacy and reliability of the detection system. It commences with the acquisition of a diverse dataset encompassing both typical and aberrant activities, reflective of potential cyber threats. Subsequently, the dataset undergoes rigorous preprocessing, including noise elimination and feature transformation, to optimize its suitability for machine learning algorithms. Feature engineering follows, wherein pertinent attributes are extracted to enhance the discriminative capacity of the models. Careful selection of appropriate algorithms, such as decision trees, random forests, or deep learning architectures like convolutional neural networks, is pivotal, considering factors like dataset size and computational resources. Through meticulous training, models learn to discern between benign and malicious activities, often employing techniques like cross-validation to bolster robustness.

Evaluation metrics, including accuracy and precision, validate model performance, alongside real-world scenario testing on independent datasets. Adversarial testing further fortifies the system, probing its resilience against malicious inputs. Continuous monitoring, retraining, and integration of feedback ensure the system evolves adeptly, adapting to emerging threats and maintaining its efficacy over time. Cyber threat detection using machine learning has become a critical area of research and development in the field of cybersecurity. Machine learning algorithms are adept at analyzing vast amounts of data to identify patterns and anomalies that may indicate a cyber threat. By leveraging techniques such as anomaly detection, supervised learning, and unsupervised learning, these systems can detect previously unknown threats and adapt to new attack vectors. For instance, supervised learning models can be trained on labeled datasets containing examples of benign and malicious activities, enabling them to classify future activities with high accuracy. Anomaly detection, which can signal potential security breaches. Additionally, the continuous learning capabilities of these models allow for the dynamic adaptation to evolving threats, enhancing the resilience of cybersecurity measures. Overall, the integration of machine learning in cyber threat detection provides a proactive approach to identifying and mitigating potential security risks, significantly improving the robustness of defense mechanisms against sophisticated cyber attacks.

5.3 Types of Testing

5.3.1 Unit testing

The unit testing may involve testing different components separately. here, we have tested the vectorizer by giving an input URL to test the feature ex- traction. It is a snippet of Python code that demonstrates how to vectorize a URL string into a numerical format for processing. The code assigns the URL "http://www.facebook.com" to a variable named 'url'. It then utilizes the 'CountVec- torizer()' function from the sklearn library to convert the URL string into a numeri- cal array. The process involves initializing the vectorizer, fitting it to the URL, and transforming the URL into numerical data, which is then printed out as an array "[[1 1 1 1]]". This technique is commonly used in machine learning and natural lan- guage processing to convert text data into a format that algorithms can work with. Unit testing involves the systematic evaluation of individual components or modules within the detection system to verify their functionality and accuracy. Each compo- nent, such as data preprocessing algorithms, feature extraction methods, or machine learning models like Support Vector Machines (SVM), undergoes rigorous testing to validate its behavior under different scenarios and edge cases.

Figure 5.3: **Unit Test Image**

5.3.2 Integration testing

In integral testing may involve testing the entire system end-to-end. So in inte- gral testing the trained SVM model along with the vectorizer is being tested and

the results are verified. Integration testing is crucial in the realm of cyber threat detection as it ensures that individual components of the system work harmoniously together to achieve effective threat identification and response. In cyber threat detection systems, integration testing involves examining the interactions between various modules, such as data collection, preprocessing, feature extraction, machine learning algorithms, and response mechanisms. This testing phase verifies the seamless integration of these components, ensuring that data flows smoothly through the system and that each module performs its designated function accurately and efficiently. It shows the output of a machine learning model evaluation in a console environment. The output includes an array and a list of strings, followed by an accuracy scoreof approximately 0.998. A detailed classification report is also presented, showing perfect precision, recall, f1-score, and support metrics for two classes labeled “bad” and “good.” The values indicate that the model has performed exceptionally well in classifying the given data. The dark background of the console highlights the text, making it easy to read the perfect scores across all categories, which suggests that the model is highly effective in its predictions.

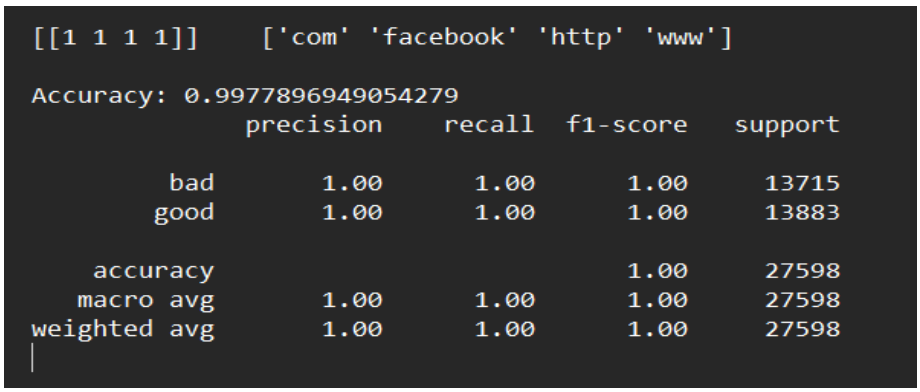


Figure 5.4: **Integration Test Image**

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed system is based on the Support Vector Machine that is used for both regression and the classification tasks. Accuracy of proposed system is done by using random forest gives the output approximately 90 to 95 percentage. Support Vector Machine gives the most accurate output when compared to the decisiontree. Support Vector Machine user the count vectorizer to extract the features ofthe URL's in the dataset and converts the features into numerical data which helps making the training easy. The by using these features extracted by the count vec- torizer the support vector machine classifies the URL's into different categories(like good,bad,etc.,) based on the features extracted.

The Support Vector Machines involves several steps that include: 1. Dataset collection: First we have to gather the data that we are going to use in model training into a dataset. 2. Data pre-processing : It refers to the cleaning of the data like eliminating the anomalies, removing the null values, dropping the duplicates etc. 3.Initializing and training the SVM: This involves the fitting the model with the dataset and finding the hyperplane in order to divide the features and classifyingif the data. 4.Model evaluation: This step involves the calculation of the metricslike accuracy, precision, recall etc. 5.Model deployment: This is the final step that involves saving the trained t=model and deploying it into a website to predict thenew and unseen links.

6.2 Comparison of Existing and Proposed System

Existing system:(Decision tree)

In the Existing system, we implemented a decision tree algorithm that predicts whether the URL is good or bad. When using a decision tree model, it gives the training dataset the accuracy keeps improving with splits. We can easily overfit the dataset and doesn't know when it crossed the line unless we are using the cross validation. The advantages of the decision tree are model is very easy to interpret we can know that the variables and the value of the variable is used to split the data. But the accuracy of decision tree in existing system gives less accurate output that is less when compared to proposed system. And also it is not suitable when we are havinga large dataset to be trained.

Proposed system:(Support Vector Machines)

Support Vector Machine gives better accuracy than the existing system that uses the decision tree. The support vector machines performs better in case of high-dimensional data without overfitting. The SVM's are very easy to handle and train in case of using the large datasets. This algorithm is memory efficient as it uses the vectors to decision function. But these models take more time when compared to the existing system. It might not perform well in case of the high overlapping datsets and also in case of datasets with inappropriate margins. Initially, data from various sources such as network traffic logs, system logs, and intrusion detection alerts are collected and preprocessed to ensure consistency and eliminate noise. Feature selection and engineering follow, aiming to identify discriminative features crucial for distinguishing between normal and malicious activities. Subsequently, the SVM model is trained using the preprocessed data, with careful consideration given to ker- nel selection and hyperparameter tuning for optimal performance. Evaluation met- rics such as accuracy, precision, and recall are employed to assess model efficacy. Integration with existing cybersecurity infrastructure is pivotal, enabling seamless communication for real-time threat detection and response.

6.3 Sample Code

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv('data.csv', encoding='latin1')
# Extracting URL data into a list or Series
url_data = data['url'].tolist()  # Assuming 'URL' is the column containing URLs
labels = data['type']  # Assuming 'Label' is the column containing labels

# Initialize CountVectorizer and transform URLs
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(url_data)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)

# Create an SVM Classifier
svm_classifier = SVC(kernel='linear', random_state=42)

# Train the model
svm_classifier.fit(X_train, y_train)

# Make predictions on the test set
predictions = svm_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")

# Print classification report
print(classification_report(y_test, predictions))
```

Output

Figure 6.1: **Sample output**

Figure 6.1 indicates the output where the user runs the program from the flask and the user gives the input URL to test the nature. This tool likely uses some form of machine learning or algorithm to assess the content of the URL and provide insights or classifications based on it. The machine learning model classifies the detected activity into different threat categories such as malware, phishing, insider threat, etc. This helps security analysts prioritize and respond to threats effectively. The output may include a risk score associated with each detected threat, indicating the severity and potential impact of the threat on the organization's security pos-ture. Higher risk scores signify more urgent threats that require immediate attention. By providing such detailed output, machine learning-based cyber threat detection systems empower organizations to proactively identify and mitigate security threats, thereby strengthening their overall cybersecurity posture. The system begins by collecting extensive data from multiple sources, including network traffic logs, system event logs, user behavior data, and external threat intelligence feeds. This data is continuously gathered to ensure up-to-date and comprehensive coverage. The collected data undergoes preprocessing to remove noise, handle missing values, and normalize different data formats. This step ensures that the data is clean and suitable for analysis. Relevant features are extracted from the preprocessed data.

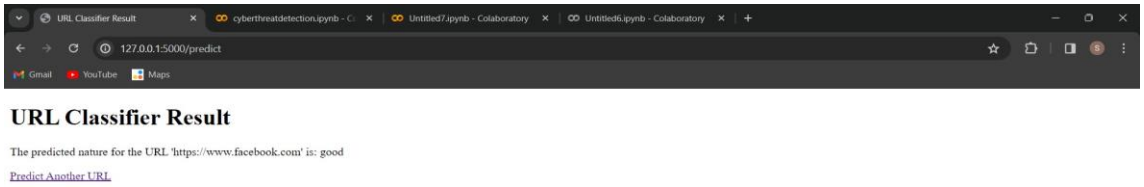


Figure 6.2: **Prediction result**

Figure 6.1 indicates the predictions on the given user input. The prediction will be the user input URL followed by it's nature whether good or bad. It depicts the results page of a web application named "URL Classifier." This application seems to evaluate and classify URLs based on predefined criteria. In the screenshot, the appli- cation has analyzed the URL "https://www.facebook.com" and determined its nature to be "good." The interface is minimalistic, featuring a plain background with optionsto classify another URL, suggesting an interactive and user-friendly design. These prediction results serve as the cornerstone for proactive defense strategies. These results typically encompass a range of crucial information essential for swift and ac- curate response to potential security breaches. They often include probability scores indicating the likelihood of malicious activity, alongside threat labels specifying the nature of the detected threat, be it malware, phishing, or other forms of cyber attacks. The system identified an unusual spike in network traffic originating from an internal IP address during off-peak hours. Using a Random Forest classifier, this anomaly was flagged as a potential Distributed Denial-of-Service (DDoS) attack. Concurrently, an Autoencoder model abnormal login patterns from a user account, it is significantly fro -m historical behaviour, suggesting a possible account compromise.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The main aim of the project is to develop a platform that helps people to check whether the URL that is going to be used is either good or bad by using machine learning algorithms. Here, the problem state can be understood from the aim of the project i.e., to overcome the unknown usage of malicious links which leads to data theft and other cyber attacks. So by using the Machine Learning algorithms detection of such links can be done which helps people in preventing cyber attacks. In con- clusion, this project cyber threat detection using machine learning helps the peopleto prevent the cyber attacks by detecting the threat before becoming a victim. This project involves the utilization of Machine Learning algorithms in order to detect the URL's by comparing them with the dataset that are used while training the algorithm. The integration of Machine learning in cyber threat detection will provides greater enhancement in the cyber safety. API user can be able to providethe input to the server can be used to get the output from the platform.

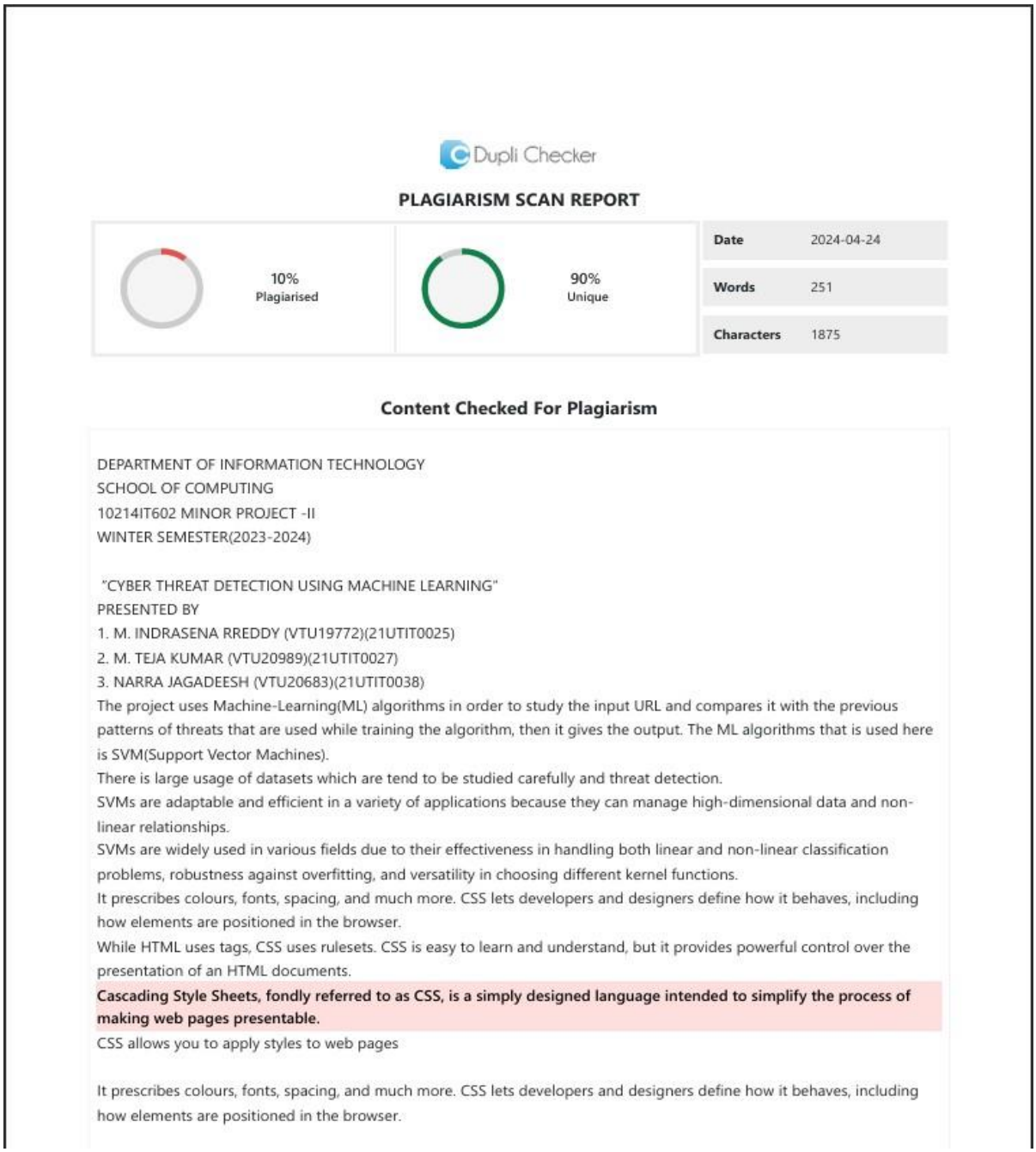
7.2 Future Enhancements

Enhanced Data Collection: This involves collection of vast number of various types of data which helps in improving the accuracy of the algorithm. It also involves data pre-processing that extracts many advanced features of the url such as URL domain,URL structure analysis and so on.

Results Enhancement: Training the model by a set of rules in order to identify the Phishing URL's based on previous patterns of the phishing URL's and their features. Along with this many other types of URL identification like Malware etc., by training the model with appropriate features and the patterns.

Chapter 8

PLAGIARISM REPORT



Dupli Checker

PLAGIARISM SCAN REPORT

10% Plagiarised

90% Unique

Date	2024-04-24
Words	251
Characters	1875

Content Checked For Plagiarism

DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
10214IT602 MINOR PROJECT -II
WINTER SEMESTER(2023-2024)

"CYBER THREAT DETECTION USING MACHINE LEARNING"
PRESENTED BY
1. M. INDRASENA RREDDY (VTU19772)(21UTIT0025)
2. M. TEJA KUMAR (VTU20989)(21UTIT0027)
3. NARRA JAGADEESH (VTU20683)(21UTIT0038)
The project uses Machine-Learning(ML) algorithms in order to study the input URL and compares it with the previous patterns of threats that are used while training the algorithm, then it gives the output. The ML algorithms that is used here is SVM(Support Vector Machines).
There is large usage of datasets which are tend to be studied carefully and threat detection.
SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and non-linear relationships.
SVMs are widely used in various fields due to their effectiveness in handling both linear and non-linear classification problems, robustness against overfitting, and versatility in choosing different kernel functions.
It prescribes colours, fonts, spacing, and much more. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.
While HTML uses tags, CSS uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML documents.
Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable.
CSS allows you to apply styles to web pages

It prescribes colours, fonts, spacing, and much more. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

Figure 8.1: **Plagiarism Report for Cyber Threat Detection**

Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source Code

```
from flask import Flask, render_template, request
import pandas as pd
import joblib
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
app = Flask(__name__)
data = pd.read_csv('minidata.csv', encoding='latin1')
url_data = data['url'].tolist()  # Assuming 'URL' is the column containing URLs
labels = data['type']  #Assuming 'type' is the column containing labels
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(url_data)
svm_classifier = SVC(kernel='linear', random_state=42)s
vm_classifier.fit(X, labels)
joblib.dump(svm_classifier, 'svm_modelfinal.joblib')
joblib.dump(vectorizer, 'count_vectorizerfinal.joblib')  # Save the CountVectorizer too
# Route for the home page
@app.route('/')
def home():
    return render_template('index.html')
# Route for prediction
@app.route('/predict', methods =['POST'])d
ef predict():
    if request.method == 'POST':
        # Load the trained model and vectorizer
        loaded_svm = joblib.load('svm_modelfinal.joblib')
        loaded_vectorizer = joblib.load('count_vectorizerfinal.joblib')
        user_input_url=request.form['url']
        # Transform the new URL using the loaded vectorizer
        url_transformed = loaded_vectorizer.transform([user_input_url])
        predicted_label=loaded_svm.predict(url_transformed)[0]
        return render_template('result.html', url=user_input_url, predicted_label=predicted_label)
if __name__ == '__main__':
    app.run(debug=True)
```

9.2 Poster Presentation

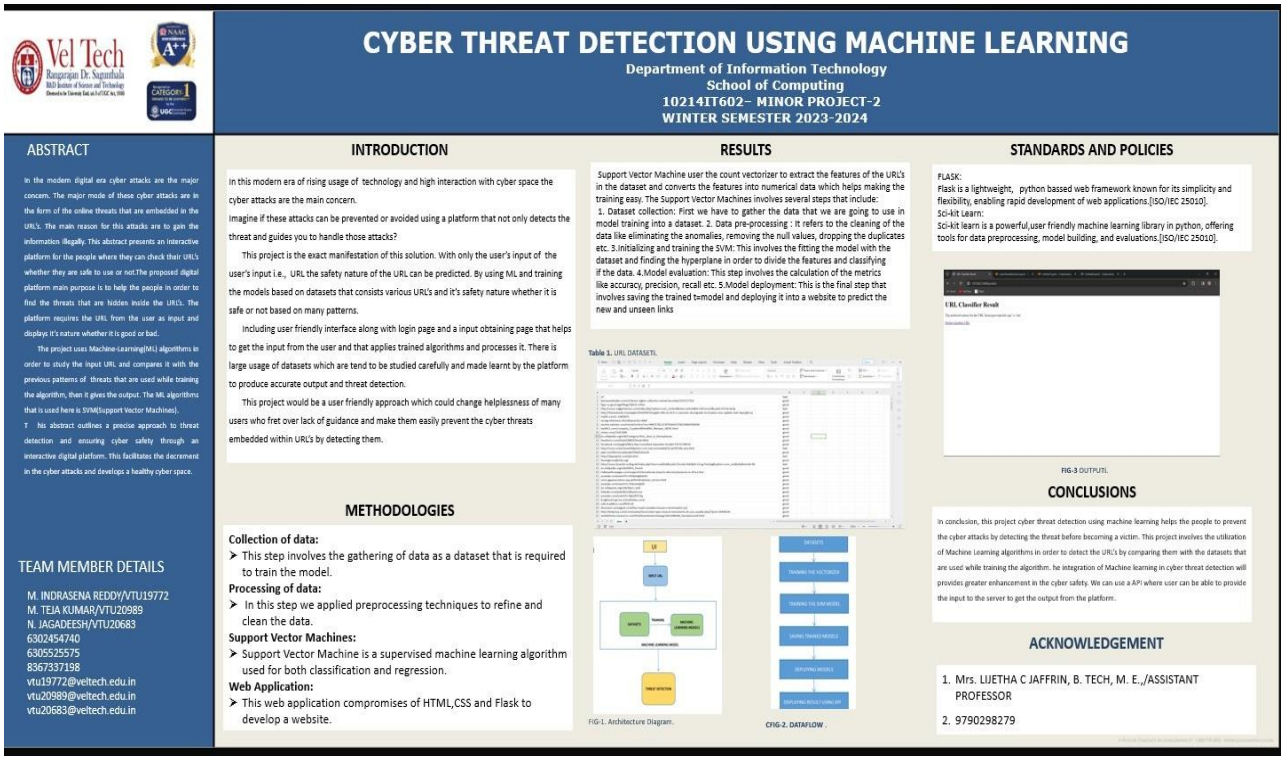


Figure 9.1: **Poster Presentation for Cyber Threat Detection**

Figure 9.1 indicates the brief description of the project in the form of poster which contains of abstract, introduction, methodology and result. This poster presentation underscores the pivotal role of machine learning in fortifying cyber de- fenses against evolving threats. By leveraging advanced algorithms and data-driven approaches, organizations can proactively identify and mitigate cyber risks, safe- guarding critical assets and infrastructure in an increasingly digitized world. This poster presentation explores the application of machine learning techniques in the realm of cyber threat detection, highlighting their effectiveness in bolstering security measures. The objective is to showcase the potential of machine learning algorithms in identifying and preempting diverse cyber threats. To discuss real-world applica- tions and case studies demonstrating the efficacy of machine learning in cyber threat detection.

References

[1] A. Alsaedi, Fuad A. Ghaleb, Faisal Saeed, Jawad Ahmad and Mohammed Alsali, "Cyber threat intelligence-based malicious url detection model using ensemble learning", Sensors, 22, no. 9: 3373, 2022.

[2] C. Catak, Ferhat Ozgur, Kevser Sahinbas, and Volkan Dörtkardeş, "Malicious url detection using machine learning", Artificial intelligence paradigms for smart cyber-physical systems. IGI global 160-180, 2021.

[3] C. Ding, "Automatic Detection of Malicious URLs using Fine-Tuned Classification Model", 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), Shenyang, China, 302-320, pp.2022.

[4] Ch. Rupa, G. Srivastava, S. Bhattacharya, P. C. Reddy, and T. R. Gadekallu, "A Machine Learning Driven Threat Intelligence System for Malicious URL Detection", Reliability and Security, Art. no. 154, Aug. 2021.

[5] Do Xuan, Cho, Hoa Dinh Nguyen, and Victor Nikolaevich Tisenko "Malicious URL detection based on machine learning", International Journal of Advanced Computer Science and Applications 11.1 (2020).

[6] E. Joshi, Apoorva, "Using lexical features for malicious URL detection–a machine learning approach", arXiv preprint arXiv:1910.06277 (2019).

[7] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour and H. Janicke, "Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks", Future Internet, vol. 12, no. 3, pp. 2020.

[8] M. Aljabri, S. S. Aljameel, R. M. A. Mohammad, S. H. Almotiri, S. Mirza, F. M. Anis, et al., "Intelligent techniques for detecting network attacks: Review and research directions", *Sensors*, vol. 21, no. 21, pp. 7070, Oct. 2021.

[9] Mohammed Alshehri, Ahed Abugabah and Sultan Almotairi Character-level word encoding deep learning model for combating cyber threatsin phishing URL detection, Volume 100, 107868, ISSN 0045-7906, 2022.

[10] S. Raja, Subrata Chowdhury and Julian L Webber Lexical features based malicious URL detection using machine learning techniques, Computer science, Volume 47, Part 1, Pages 163-166, 2021.