**A Project Report on**

**Healthcare Chatbot using Machine Learning**

submitted in partial fulfillment for the award of

**Bachelor of Technology**

in

**Data Science**

by

**U. Nagendra Obuldasu (Y20ADS428) R. Naga Nandini (Y20ADS425)**

**R. Sumanth (Y20ADS424) N. Chandra Vamsi (L21ADS402)**



Under the guidance of

**Asst. Prof. M. Rishitha**

Department of CB, DS & AIML

**Bapatla Engineering College**

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

**BAPATLA – 522 102, Andhra Pradesh, INDIA**

**2023-2024**

**Department of**

**CB, DS & AIML**



**CERTIFICATE**

This is to certify that the project report entitled **Healthcare chatbot using Machine learning** that is being submitted by U. Nagendra Obuldasu (Y20ACB428), R. Naga Nandini (Y20ADS425), R. Sumanth(Y20ADS424) and N. Chandra Vamsi(L21ADS402) in partial fulfillment for the award of the Degree of Bachelor of Technology in Cyber Security to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

**Signature of the Guide Signature of the HOD**

**M. Rishitha V. Chakradhar**

**Asst. Prof. Prof. & Head**

**DECLARATION**

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

**U. Nagendra Obuldasu (Y20ADS428)**

**R. Naga Nandini (Y20ADS425)**

**R. Sumanth (Y20ADS424)**

**N. Chandra Vamsi (L21ADS402)**

**ACKNOWLEDGEMENT**

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **M. Rishitha, Asst. Prof.,** Department of CB, DS & AIML, for his valuable and inspiring guidance, comments, suggestions, and encouragement.

We extend our sincere thanks to **V. Chakradhar,** Prof. & Head of the Dept. of CB, DS & AIML for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal Dr. **Sk. Nazeer** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Leela Kumari, Asst.** Prof., Dept. of CB, DS & AIML for his helpful suggestions in presenting this document.

We would like to express our sincere thanks to **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

**U. Nagendra Obuldasu (Y20ADS428)**

**R. Naga Nandini (Y20ADS425)**

**R. Sumanth (Y20ADS424)**

**N. Chandra Vamsi (L21ADS402)**

**Table of Contents**

[ABSTRACT iv](#_Toc164320983)

[LIST OF TABLES vii](#_Toc164320984)

[LIST OF FIGURES viii](#_Toc164320985)

[LIST OF SYMBOLS AND OBREVATIONS ix](#_Toc164320986)

[1. INTRODUCTION 1](#_Toc164320987)

[1.1 Purpose 2](#_Toc164320988)

[1.2 Literature Survey 2](#_Toc164320989)

[1.2.1 Malicious URL Detection and Analysis using Machine Learning: 2](#_Toc164320990)

[1.2.2 Phishing Attacks Detection A Machine Learning: 2](#_Toc164320991)

[1.3 Existing System 3](#_Toc164320992)

[1.4 Disadvantages of using Blacklist 4](#_Toc164320993)

[1.5 Disadvantages of using Machine Learning Algorithms 5](#_Toc164320994)

[1.5 Proposed System 6](#_Toc164320995)

[1.5 Advantages 6](#_Toc164320996)

[1.5.1 User-friendly Interface: 8](#_Toc164320997)

[1.5.2 High Accuracy: 8](#_Toc164320998)

[1.5.3 Blacklist Integration: 8](#_Toc164320999)

[1.6 Scope 8](#_Toc164321000)

[2. SYSTEM ENVIRONMENT 9](#_Toc164321001)

[2.1 Technologies Used 9](#_Toc164321002)

[2.2 Hardware Requirements 9](#_Toc164321003)

[2.3 Software Requirements 9](#_Toc164321004)

[2.4 Libraries Used 9](#_Toc164321005)

[3. SYSTEM DESIGN 11](#_Toc164321006)

[4. ALGORITHMS 12](#_Toc164321007)

[5. MODULES AND WORK PLAN 13](#_Toc164321008)

[5. IMPLEMENTATION 14](#_Toc164321009)

[6. SYSTEM TESTING 15](#_Toc164321010)

[7. LIMITATIONS AND FUTURE SCOPE 16](#_Toc164321011)

[8. CONCLUSION 17](#_Toc164321012)

[8. **Bibliography** 18](#_Toc164321013)

LIST OF TABLES

No table of figures entries found.

LIST OF FIGURES

No table of figures entries found.

LIST OF SYMBOLS AND OBREVATIONS

1. INTRODUCTION

In today’s digital era, the internet has become an integral part of our lives, providing a vast array of information organizations alike. Malicious URLs refer to web addresses that are designed to deceive users and initiate harmful activities, such as phishing attacks, malware distribution, or identity theft.

To combat these growing cybersecurity challenges, the proposed system “Malicious URL Detection Using Random Forest and Blacklist” leverages the Random Forest algorithm, a machine learning technique renowned for its ability to handle complex classification tasks. By training the model on a diverse dataset of known malicious and benign URLs, the Random Forest algorithm can learn to recognize patterns and distinguish between the two categories effectively. This allows it to identify potential threats accurately and minimize false positives, ensuring a high level of accuracy in the detection process.

In addition to the Random Forest algorithm (RF), the system incorporates a blacklist of known malicious URLs. This blacklist contains URLs that have been previously identified as malicious through various sources, such as threat intelligence feeds, security vendors, and user reports. By cross-referencing incoming URLs against this blacklist, the system can swiftly identify and block access to known malicious URLs, providing an additional layer of protection.

1.1 Purpose

The purpose of this project is to integrate ML model with the blacklist to increase the efficiency and accuracy of the URL detection with time consideration. This project aims to provide individuals and organizations with a reliable defence against malicious URLs and mitigate the risks associated with cyber threats in the digital landscape.

1.2 Literature Survey

1.2.1 Malicious URL Detection and Analysis using Machine Learning:

Authors- Dr.Upendra shetty, Anusha patil, and Mohana, Published in: 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT) :This paper provides details about the reason for choosing machine learning models instead of backlist. The authors used machine learning algorithms such as random forest classifier, light GBM classifier, and XGboost classifier used to categorize URLs into Phishing, Benign, Defacement, and Malware. Here they provided the differentiation between the three machine learning models in classifying and detecting the URL comparing their accuracies. Here, they used only machine learning techniques in detecting the URLs. Here they used Random Forest algorithm for classification.

1.2.2 Phishing Attacks Detection A Machine Learning:

Authors- Fatima Salahdine, Zakaria El Mrabet, Naima Kaabouch: This paper provides an overview about detecting Phishing emails with efficient machine learning models. Here they used collection of different features extracted from URL and using those features they have trained the model. The features extracted are SSL certificate, certificate authority, blacklist keywords, redirection URL, hiding links, clear Ip address, sender email address etc... up to ten features are extracted from URLs.

The three classification techniques used are, namely Support Vector Machine (SVM), logistic regression (LR), and artificial neural network (ANN). They trained the model with four thousand real phishing emails. The ANN model got the highest accuracy rate of 94.5% and allows fast and accurate phishing attacks detection.

1.3 Existing System

The existed malicious URL detection systems are built using Machine learning techniques to detect the Phishing or malicious URLs more accurately when compared to the techniques such as Blacklisting, Whitelisting.

The machine learning algorithms such as Random Forest, Logistic Regression, Support Vector Machine (SVM) are used to build the classification model to classify the URLs and their types like defacement URL, phishing URL, malicious URL or benign. The Blacklist technique works as a database where the known malicious URLs are stored, and it classifies the URL as malicious if it contains the input URL.

1.4 Disadvantages of using Blacklist

1. **Limited to known threats - only detects the URLs that have already been detected and added to the list:**

The system can only identify malicious URLs that have been previously identified and catalogued in its list of known threats. In other words, it relies solely on historical data or pre-existing knowledge of threats to detect malicious URLs. Any new or previously unseen malicious URLs may not be detected by this system since they are not part of its known threats list.

1. **False negatives - indicates malicious URLs as legitimate:**

The detection system fails to recognize a malicious URL as malicious and incorrectly identifies it as safe or legitimate. In other words, it misses detecting URLs that are harmful, leading to a false sense of security. False negatives are a significant concern in cybersecurity because they can result in users unknowingly accessing malicious websites or content, putting their data and systems at risk.

1. **Evasion techniques-methods used by attackers to bypass or evade detection by security systems, including those designed to detect malicious URLs:**

This statement refers to the strategies and tactics employed by cyber attackers to circumvent or trick security systems into not detecting their malicious activities, including the use of malicious URLs. Evasion techniques can include various methods such as obfuscation, encryption, polymorphism, and social engineering. Attackers continuously develop and refine evasion techniques to avoid detection by security measures, making it challenging for defenders to effectively protect against cyber threats.

1.5 Disadvantages of using Machine Learning Algorithms

1. **False negatives - indicates legitimate as malicious:**

The system incorrectly identifies legitimate URLs as malicious. False negatives occur when the detection mechanism fails to recognize actual threats, leading to benign URLs being flagged as malicious. This can be problematic as it can cause unnecessary disruption or blocking of legitimate websites or content, potentially impacting user experience and productivity.

1. **Delayed response - to classify even known malicious URLs it takes more time:**

The system experiences delays in responding to threats, even when they are already known and identified as malicious. Despite having prior knowledge of certain malicious URLs, the system takes longer than expected to classify and act upon them. Delayed response times can hinder the effectiveness of threat mitigation efforts, allowing malicious URLs to remain active for longer periods and potentially causing more damage.

1. **Resource-intensive training - to classify URL every time it takes more power:**

Here, it's noted that the process of training the system to classify URLs requires significant computational resources. Training machine learning models or algorithms to accurately classify URLs can be computationally intensive, consuming considerable processing power, memory, and time. This resource-intensive training process can pose challenges in terms of scalability, cost, and operational efficiency, especially when dealing with large datasets or frequent updates to the classification models.

1.6 Proposed System

The Proposed system is the integration of Machine learning model with Blacklist. This system passes the input URL to blacklist firstly. If the url is not present in blacklist, the URL is passed to the machine learning model and the model will classify either URL is safe or not. If the machine learning model classifies the URL as Malicious, means not safe, then we add that malicious URL to the blacklist, which reduces the time complexity for detecting the same URL again in future.

1.7 Advantages

1. **Enhanced accuracy - machine learning can be used to detect known and new threats, blacklist used quickly identify known malicious URLs:**

This statement suggests that machine learning techniques are employed to improve the accuracy of threat detection. Machine learning algorithms can analyze patterns and characteristics of known malicious URLs, as well as learn from new data to detect emerging threats. Additionally, the use of blacklists, which are lists of known malicious URLs, can help quickly identify and block such URLs, further enhancing the accuracy of threat detection.

1. **Reduced false positives and false negatives - both techniques balance false positives and false negatives:**

Here, it's highlighted that the employed techniques aim to minimize both false positives and false negatives. False positives occur when benign URLs are incorrectly identified as malicious, while false negatives occur when malicious URLs are incorrectly identified as benign. Balancing these two types of errors is crucial for maintaining the effectiveness of the detection system and avoiding unnecessary disruptions or missed threats.

1. **Scalability - able to handle a large amount of data:**

This statement indicates that the system is designed to efficiently process and analyse large volumes of data. In the context of threat detection, scalability ensures that the system can effectively manage the ever-growing volume of URLs and data associated with cyber threats without experiencing performance degradation or resource constraints.

1. **Robustness - creating a system that is less vulnerable to evasion techniques:**

Here, the focus is on building a resilient system that is capable of withstanding attempts by attackers to bypass or evade detection. This involves implementing security measures and techniques that make it difficult for attackers to exploit vulnerabilities or weaknesses in the system. By enhancing robustness, the system becomes more resilient against evasion techniques and better equipped to defend against emerging threats.

1. **Adaptability - ability to adapt to new threats:**

This statement underscores the importance of flexibility and responsiveness in the face of evolving cyber threats. An adaptable system can quickly adjust its detection capabilities and strategies to effectively identify and mitigate new and emerging threats. This may involve updating machine learning models, blacklists, or detection algorithms based on the latest threat intelligence and trends in cyber-attacks.

1.5.1 User-friendly Interface:

The user interface should be intuitive and easy to use, allowing users to submit URLs for evaluation and view the results in a clear and understandable format.

1.5.2 High Accuracy:

The system should achieve an elevated level of accuracy in distinguishing between malicious and benign URLs. Users expect the system to minimize false positives (legitimate URLs flagged as malicious) and false negatives (malicious URLs not detected).

1.5.3 Blacklist Integration:

The system should incorporate a regularly updated blacklist of known malicious URLs obtained from reputable cybersecurity sources. The integration of the blacklist should be seamless and provide up-to-date threat intelligence.

1.6 Scope

The project's scope is primarily centred around the development and implementation of an accurate and efficient malicious URL detection system using Random Forest algorithms and blacklisting techniques. By defining the scope, the project aims to establish clear boundaries and deliver a focused solution that addresses the key objectives and requirements of the system.

2. SYSTEM ENVIRONMENT

2.1 Technologies Used

* The versatile programming language well known for its simplicity and readability, nothing but **PYTHON** was used.
* The standard markup language **HTML** was used for documents designing and to be displayed in a web browser.

2.2 Hardware Requirements

* A **laptop/desktop** that having descent specifications and in good running condition.
* That laptop must contain a minimum of **4GB RAM** and a descent **HDD/SDD** (storage) capability.

2.3 Software Requirements

* To execute our project, you need **pycharm**.
* You will get better experience with the OS on or above **windows 10.**

2.4 Libraries Used

1. **Scikit Learn:** A powerful machine learning library in Python, to implement various algorithms for tasks such as classification, regression, clustering, and dimensionality reduction.
2. **Pandas:** A popular Python library, offering powerful tools for handling structured data through its DataFrame and Series data structures.
3. **Numpy:** A fundamental library for scientific computing in Python, provides support for powerful array operations, mathematical functions, linear algebra, and random number generation, serving as a backbone for many numerical computing tasks.
4. **Pickle:** A module in Python's standard library, facilitates the serialization and deserialization of Python objects, enabling seamless data persistence and interchangeability across different sessions or environments.
5. **Flask:** A lightweight and flexible web framework in Python, empowers developers to build web applications quickly and efficiently by providing essential tools and libraries for routing, templating, and interacting with databases.
6. **Ipaddress:** It provides classes and functions for working with IP addresses and networks, offering convenient methods for parsing, manipulating, and validating IP addresses, as well as performing subnet calculations and network operations.
7. **7. Tldextract:** A Python library, offers a convenient way to parse domain names and extract the top-level domain (TLD), domain name, and subdomains accurately, facilitating tasks such as domain parsing, data analysis, and web scraping.

3. SYSTEM DESIGN

4. ALGORITHMS

4.1 Random Forest Alforithm

Random Forest is a popular ensemble learning technique used for classification and regression tasks. It operates by constructing multiple decision trees during the training phase and outputting the mode (classification) or mean prediction (regression) of the individual trees.

1. **Bootstrapping (Random Sampling with Replacement):**

Random Forest starts by creating multiple random subsets of the training dataset with replacement. Each subset is used to train an individual decision tree.

Randomly select 'n' samples from the training dataset with replacement to create 'm' subsets, where 'n' is the size of the training dataset and 'm' is the number of decision trees in the forest.

1. **Decision Tree Construction:**

For each subset, a decision tree is constructed using a subset of features chosen randomly at each node. This randomness helps in reducing correlation among the trees and leads to a more diverse set of classifiers.

At each node of the decision tree, randomly select 'k' features from the total 'p' features available in the dataset, where 'k' is typically √p (the square root of the total number of features). Use these 'k' features to find the best split.

1. **Voting (Classification) or Averaging (Regression):**

During prediction, each tree in the forest generates a class prediction (for classification) or a numeric prediction (for regression). The final prediction is determined by aggregating the predictions of all the trees. For classification, the mode (most frequent class) is taken as the final prediction, while for regression, the mean of all predictions is calculated.

* **Classification:** For each test sample, let \( c\_1, c\_2, ..., c\_m \) be the class predictions of the individual decision trees. The final prediction is the mode of the predictions: \( \text{Mode}(c\_1, c\_2, ..., c\_m) \).
* **Regression:** For each test sample, let \( r\_1, r\_2, ..., r\_m \) be the numeric predictions of the individual decision trees. The final prediction is the mean of the predictions: \( \text{Mean}(r\_1, r\_2, ..., r\_m) \).

Random Forest is highly effective in practice due to its ability to handle high-dimensional data, handle missing values, and reduce overfitting compared to individual decision trees. Additionally, it's relatively resistant to noise and outliers in the data.

4.2 Light BGM Classifier

LightGBM (Light Gradient Boosting Machine) is a gradient boosting framework developed by Microsoft that is optimized for large-scale datasets and provides high-performance machine learning algorithms for classification, regression, and ranking tasks. It's known for its efficiency, speed, and accuracy.

LightGBM belongs to the family of gradient boosting algorithms, which work by sequentially adding weak learners (decision trees in this case) to an ensemble, with each tree correcting the errors of its predecessors. LightGBM specifically utilizes a technique called gradient-based one-side sampling (GOSS) to reduce memory usage and increase training speed.

1. **Tree Building:**

LightGBM grows trees vertically (leaf-wise) rather than horizontally (level-wise), which means that it splits nodes that yield the highest reduction in the loss function. This approach leads to deeper trees compared to depth-wise tree growth algorithms like traditional gradient boosting (e.g., XGBoost).

LightGBM optimizes a user-defined objective function by iteratively fitting weak learners (decision trees) to minimize the loss. The objective function can be tailored based on the specific task (e.g., binary classification, regression).

1. **Leaf-wise Splitting:**

In LightGBM, each node is split based on the maximum reduction in the loss function, resulting in a more precise but potentially overfit model. To control overfitting, LightGBM employs regularization techniques like max\_depth, min\_child\_samples, and min\_child\_weight.

At each node of the tree, LightGBM calculates the gradient of the loss function with respect to the prediction, then splits the node in the direction that maximally reduces the loss.

1. **Gradient-Based One-Side Sampling (GOSS):**

LightGBM employs a sampling method called GOSS to speed up the training process. GOSS keeps the instances with large gradients (which contribute more to the model training) while randomly subsampling instances with small gradients. This allows for faster convergence during training without sacrificing performance.

GOSS subsamples instances with small gradients to reduce memory usage and accelerate training. The formula for determining which instances to keep or discard during sampling depends on the gradients of the loss function.

1. **Exclusive Feature Bundling (EFB):**

LightGBM can bundle exclusive features together during training, reducing memory usage and speeding up computation. EFB groups features that rarely appear together, reducing the number of feature combinations that need to be considered during tree construction.

LightGBM's efficient implementation and optimization techniques make it suitable for handling large-scale datasets with millions of instances and features. It often outperforms other gradient boosting frameworks in terms of speed and accuracy, especially on tabular data.

4.3 XGboost classifier

XGBoost (Extreme Gradient Boosting) is a powerful and widely used machine learning algorithm for classification and regression tasks. It belongs to the family of ensemble learning techniques and is based on the gradient boosting framework.

XGBoost is an ensemble learning method that builds a strong predictive model by combining the predictions of multiple weak models, typically decision trees. It's known for its scalability, speed, and performance on structured/tabular data. XGBoost uses a gradient boosting framework to iteratively improve the predictive performance of the model.

1. **Decision Tree Construction:**

XGBoost builds decision trees sequentially, with each tree aiming to correct the errors made by the previous trees. It starts with an initial prediction (e.g., the mean of the target variable) and then builds subsequent trees to minimize the residual errors.

At each iteration, XGBoost constructs a decision tree that splits the data into regions to minimize the loss function. The split is determined by finding the optimal value of a feature that maximally reduces the loss.

1. **Gradient Boosting:**

During each iteration, XGBoost calculates the gradients of the loss function (e.g., mean squared error for regression, logistic loss for classification) with respect to the predictions. It then fits a decision tree to the negative gradient of the loss function, effectively reducing the residual error.

1. **Regularization:**

To prevent overfitting, XGBoost incorporates regularization techniques such as shrinkage (learning rate), maximum depth of trees, minimum child weight, and subsampling of training instances and features. These regularization parameters help control the complexity of the model and improve its generalization performance.

1. **Learning Rate:**

The learning rate (eta) controls the step size at each iteration of gradient descent. A lower learning rate makes the model more conservative, while a higher learning rate leads to faster convergence but may result in overfitting.

1. **Objective Function:**

XGBoost optimizes a user-defined objective function (e.g., mean squared error for regression, logistic loss for binary classification) by iteratively fitting weak learners (decision trees) to minimize the loss. The objective function typically consists of two parts: the loss function and a regularization term.

3. \*\*Gradient Calculation:\*\*

XGBoost calculates the first and second-order gradients of the loss function with respect to the predictions. The first-order gradient represents the direction of steepest descent, while the second-order gradient provides information about the curvature of the loss function.

4. \*\*Regularization:\*\*

XGBoost incorporates regularization terms into the objective function to penalize complex models and encourage simplicity. Regularization parameters such as gamma, lambda, and alpha control the complexity of the model and prevent overfitting.

XGBoost's combination of gradient boosting, regularization, and optimization techniques makes it a versatile and powerful algorithm for a wide range of machine learning tasks. It's widely used in industry and academia and has won numerous machine learning competitions for its outstanding performance and efficiency.

5. MODULES AND WORK PLAN

URL Classification:The system accepts URLs as input and classifies them as either malicious or benign using the trained Random Forest classifier.URL classification is based on features extracted from both the URL itself and the blacklist of known malicious URLs.

Blacklist Integration:The system incorporates a regularly updated blacklist of known malicious URLs obtained from reputable cybersecurity sources.The blacklist provides valuable threat intelligence, enhancing the accuracy of URL classification.

Feature Extraction:The system extracts relevant features from URLs to aid in URL classification.Domain-based features, path-related features, and content-based features are considered to identify patterns indicative of malicious URLs.

5. IMPLEMENTATION

Collect the dataset

2. Explore the dataset

3. Store malicious urls to blacklist

4. Extract features from the collected dataset 5.Train and validate the model

6. Building model

7. Test the model with dataset 8.Concluding the accuracy rate

9.Integrating blacklist with random forest algorithm 10.Building Graphical User Interface

6. SYSTEM TESTING

7. LIMITATIONS AND FUTURE SCOPE

8. CONCLUSION

# **Bibliography**

|  |  |
| --- | --- |
| [1] | Fathima Salahdine, Z. E. M. and N. K. , “Phishing Attacks Detection A Machine Learning-Based Approach,” p. 6, 2021. |
| [2] | R. Mohammad, F. T. and L. M. , “Intelligent rule-based phishing websites classification,” *IET Inf. Secur.,* p. 153–160, 2014. |
| [3] | Upendra Shetty D R, Anusha Patil and Mohana, “Malicious URL Detection and Classification Analysis using Machine Learning Models,” *Proceedings of the International Conference on Intelligent Data Communication Technologies and Internet of Things,* p. 7, 2023. |
| [4] | D. Ranganayakulu, “Detecting Malicious URLs in E-mail – An Implementation,” *AASRI Procedia,* vol. 4, pp. 125-131, 2013. |
| [5] | M. F. A. Razak, “Comparative Analysis of Machine Learning Classifiers for Phishing Detection,” *6th International Conference on Informatics and Computational Sciences (ICICoS),* pp. 84-88, 2022. |
| [6] | M. Atari, “A Machine-Learning Based Approach for Detecting Phishing URLs,” *International Conference on Intelligent Data Science Technologies and Applications (IDSTA),* pp. 82-88, 2022. |
| [7] | E. Nowroozi, “An Adversarial Attack Analysis on Malicious Advertisement URL Detection Framework,” *IEEE Transactions on Network and Service Management,* 2022. |
| [8] | R. J. Franklin, “Anomaly Detection in Videos for Video Surveillance Applications using Neural Networks,” *International Conference on Inventive Systems and Control (ICISC),* pp. 632-637, 2020. |