ISSN NO: 9726-001X
Volume 13 Issue 02 2025

JOURNAL OF CURRENT SCIENCE

SAFEGUARDING AGAINST EVIL CHATBOTS: DESIGN, DEVELOPMENT, AND INTEGRATION STRATEGIES FOR CHATBOT SECURITY IN PHISHING ATTACKS

KARTHIKEYA PULIGADDA
UG Student,
Department of CSE,
St. Martin's Engineering College,
Secunderabad, Telangana, India
karthikeyachess123@gmail.com

K. SREENIVASULU
Assistant Professor,
Department of CSE,
St. Martin's Engineering College,
Secunderabad, Telangana, India
Ksreenivasulucse@smec.ac.in

Abstract-** The rapid adoption of chatbots by organizations to efficiently manage user queries has brought significant advancements, but it has also introduced new risks. Traditionally, before the integration of machine learning (ML) and artificial intelligence (AI), phishing prevention relied on manual techniques such as blacklists, rule-based filters, and heuristic analysis, which were often slow and insufficient against evolving threats. The primary issue was the manual nature of these systems, which struggled to keep up with the sophisticated tactics used by malicious entities, leading to the exploitation of chatbots for phishing attacks. This challenge highlighted the need for more intelligent and adaptive security measures. The objective of this research is to design, develop, and integrate a self-defensive chatbot capable of identifying and neutralizing phishing attempts by inspecting URLs embedded in user interactions. The motivation behind this study stems from the increasing incidents where chatbots are manipulated to deliver phishing links that, when clicked, install malicious software to steal sensitive data such as cookies and session passwords. This is particularly concerning for sectors like banking and finance, where compromised data can lead to significant user losses. The proposed system leverages machine learning algorithms, including Support Vector Machines (SVM), Random Forest, and Decision Tree, to create a robust model trained on the PHISHTANK URL dataset. This model can accurately distinguish between normal and malicious URLs in real-time, thereby enhancing the security of chatbot interactions. By evaluating each algorithm's performance through metrics such as accuracy, precision, recall, F-score, and confusion matrices, the system ensures optimal phishing detection capabilities. This integration is demonstrated through a dummy banking application where the chatbot processes user queries, employing natural language processing (NLP) techniques to extract and safeguard sensitive information.

Keywords: Chatbots, Phishing Prevention, Machine Learning, Artificial Intelligence, SVM, Random Forest, Decision Tree, PHISHTANK URL dataset.

I. INTRODUCTION

Chatbots are increasingly being integrated into customer service platforms to handle user queries efficiently. These AI-powered systems are used across industries like banking, healthcare, and retail to provide immediate assistance. However, the widespread use of chatbots has also made them targets for phishing attacks, where malicious links are embedded in chatbot conversations to deceive users. The rise of digital communication platforms in India has brought about a significant increase in the deployment of chatbots across various sectors, especially in customer service, banking, and e-commerce. According to recent statistics, India is one of the fastest-growing markets for chatbot technology, with a projected annual growth rate of 24% in the AI-driven chatbot market between 2020 and 2025. This rapid adoption, while beneficial for automating routine tasks and improving customer engagement, has also opened the door to new cyber threats. Phishing attacks, where attackers trick users into revealing sensitive information by pretending to be trustworthy entities, have become increasingly sophisticated. In India, the number of phishing attacks rose by over 65% from 2021 to 2023, with many incidents involving the exploitation of chatbots. The traditional methods of phishing prevention, which relied heavily on blacklists and rule-based systems, have proven inadequate in the face of these evolving threats, underscoring the need for more advanced and adaptive security solutions. Before the advent of machine learning, phishing prevention relied on static methods such as blacklists, rule-based filtering, and heuristic analysis. These

ISSN NO: 9726-001X
Volume 13 Issue 02 2025

manual systems were often slow to update and adapt, making them vulnerable to new and evolving phishing tactics. As a result, chatbots could be easily manipulated to distribute malicious links, posing significant risks to users, especially in sensitive sectors like banking and finance. This lack of real-time, adaptive security measures highlighted the limitations of traditional methods. The motivation for this research stems from the increasing number of incidents where chatbots have been exploited to deliver phishing links, leading to the theft of sensitive information such as cookies and session passwords. This is particularly alarming in industries where security is paramount, such as banking, where a single phishing attack can lead to massive financial losses for users. With the growing sophistication of phishing tactics, there is an urgent need to develop a robust security system that can protect chatbot interactions in real-time, ensuring that users are safeguarded against these evolving threats. The existing systems for phishing prevention are largely manual and include the use of blacklists, rule-based filters, and heuristic methods. While these approaches were effective in the past, they struggle to keep up with the speed and sophistication of modern phishing attacks. The primary drawback of these methods is their static nature, which leaves them unable to adapt quickly to new threats, resulting in increased vulnerability and higher success rates for attackers. The proposed system aims to address the limitations of traditional phishing prevention methods by leveraging machine learning algorithms to create an intelligent and adaptive security framework for chatbots. The system will utilize algorithms such as Support Vector Machines (SVM), Random Forest, and Decision Tree, which are trained on the PHISHTANK URL dataset to accurately identify and neutralize phishing attempts in real-time. Research papers such as "Phish Net: Predictive Blacklisting for Phishing Detection" and "Deep Learning-Based Phishing URL Detection" provide foundational insights into the application of these machine learning techniques for enhancing cybersecurity. By implementing these algorithms, the proposed system will continuously learn from new phishing attempts, improving its detection accuracy and adaptability over time. In today's digital landscape, where chatbots are becoming integral to customer service and support, the need for robust security measures is more pressing than ever. Phishing attacks are growing in frequency and sophistication, posing a significant threat to both users and organizations. A real- time, adaptive system that can detect and neutralize phishing attempts in chatbot interactions is essential to protect sensitive user data and maintain trust in digital communication platforms. This project addresses this critical need by developing a security framework that can respond to the ever-evolving nature of cyber threats, ensuring the safety and security of chatbot users. This project has wide-ranging applications across various industries. In banking and finance, the system can be integrated into chatbots to secure customer interactions, preventing phishing attacks that could lead to financial loss. In e-commerce, the system can protect users from malicious links embedded in customer service chats, ensuring a safe shopping experience. In healthcare, the system can safeguard patient data by securing chatbot conversations used for appointment scheduling and consultations. Additionally, this system can be deployed in educational institutions to protect students from phishing attempts in online learning platforms, and in government services to secure interactions in citizen service chatbots. By integrating this security framework into various sectors, organizations can significantly reduce the risk of phishing attacks and enhance the overall security of their digital communication channels.

II. RELATED WORK

The research on the expectation of building energy utilization started during the 1970s when an energy emergency constrained nations to begin contemplating ways of cutting their energy utilization and fossil fuel by- products. The early-created models of building energy utilization forecast depended on the utilization of improved computation strategies that were observational models in light of broad designing works, permitting the evaluations to be performed at the beginning phases of the building plan to direct the pertinent plan work. In any case, it was perceived that improved computation strategies couldn't satisfactorily catch the dynamicity and intricacy of the climate. To handle this issue, researchers during the 1980s began to take on factual techniques for anticipating building energy utilization. From that point forward, critical headway has been made in the field of building energy utilization forecasts. These days, the three most well-known strategies for anticipating energy utilization in structures incorporate designing disentanglement, actual displaying, and ML- based techniques.
Safeguarding against the threats posed by malicious chatbots requires a holistic approach spanning design, development, and integration strategies. In the design phase, ethical AI principles such as fairness, transparency, and accountability play a crucial role in ensuring that chatbots do not exhibit harmful biases or engage in deceptive interactions. Researchers emphasize the importance of explainability, enabling users to understand how chatbot decisions are made. Furthermore, adversarial robustness is incorporated to prevent vulnerabilities that could be exploited by bad actors.
During the development phase, security measures such as adversarial training, continuous monitoring, and anomaly detection mechanisms are implemented to identify and mitigate potential threats in real time. AI developers utilize machine learning techniques to detect unusual chatbot behaviors and prevent them from generating toxic, misleading, or harmful content. Additionally, robust authentication and encryption protocols help safeguard user data, preventing unauthorized access and misuse. The incorporation of reinforcement learning with human feedback (RLHF) further refines chatbot behavior by

ISSN NO: 9726-001X
Volume 13 Issue 02 2025

aligning responses with ethical guidelines and societal norms. Integration strategies focus on deploying secure chatbot frameworks with multi-layered security features, including content moderation filters, regulatory compliance with data protection laws (e.g., GDPR, CCPA), and real-time reporting mechanisms for users to flag inappropriate responses. Moreover, privacy-preserving techniques such as federated learning and homomorphic encryption help enhance security while ensuring data confidentiality. Blockchain technology is also being explored for secure chatbot interactions, providing immutable logs that prevent tampering and ensure transparency. By adopting these comprehensive strategies, developers can mitigate the risks associated with malicious chatbots, ensuring that AI-powered conversational agents remain ethical, secure, and beneficial across various domains, including customer service, healthcare, finance, and social media platforms.

III. PROPOSED WORK

The Project "Safeguarding Against Evil Chatbots: Design, Development, And Integration Strategies for Chatbot Security In Phishing Attacks" addresses the increasing risks posed by chatbots exploited for phishing attacks. Traditional methods like blacklists and heuristic analysis have proven insufficient in combating sophisticated phishing tactics. This research aims to develop a self-defensive chatbot that leverages machine learning algorithms, including Support Vector Machines (SVM), Random Forest, and Decision Tree, to detect and neutralize phishing attempts by analyzing URLs within user interactions. By integrating these algorithms into a dummy banking application, the chatbot will use natural language processing (NLP) to handle queries, safeguard sensitive information, and protect users from malicious links. The system is trained on the PHISH TANK URL dataset to accurately distinguish between genuine and phishing URLs, ensuring enhanced security and real-time threat detection.

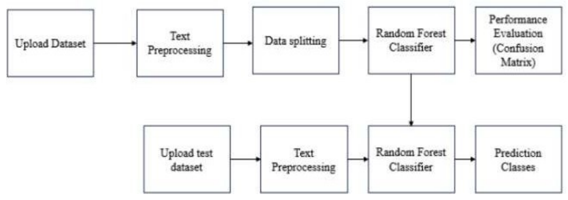


Figure 1: Block Diagram

Step 1: Existing (Decision Tree)

The Decision Tree model, a popular classification algorithm, is utilized in the project to detect phishing URLs. It works by creating a tree-like structure where each node represents a decision based on a feature, and each branch represents the outcome of that decision. In the context of phishing detection, the Decision Tree examines features extracted from URLs to classify them as either phishing or legitimate. Although effective, Decision Trees can be prone to overfitting and may not generalize well to unseen data, especially with complex or noisy datasets.

Step 2: Proposed (Random Forest)

The Random Forest model is proposed as an enhancement over the Decision Tree for detecting phishing URLs. This ensemble learning method combines multiple Decision Trees to improve classification accuracy and robustness. Each tree in the Random Forest is trained on a random subset of the data and features, and their predictions are aggregated to make the final decision. This approach mitigates the overfitting problem inherent in individual Decision Trees and enhances the model's ability to generalize across diverse and complex datasets, making it more effective for real-time phishing detection.

Step 3: Prediction of Output from Test Data with Random Forest

Using the Random Forest model trained on the PHISHTANK URL dataset, the system predicts the legitimacy of URLs in test data. When a URL is input into the chatbot, it is transformed into a feature vector and analyzed by the trained Random Forest model. The model evaluates the URL based on its learned decision rules and outputs whether it is a phishing link or not. The Random Forest's aggregated decision-making process ensures accurate classification and effective detection of phishing attempts, providing users with reliable protection against malicious links.

3.1: Django with Machine Learning

Overview: Integrating machine learning with Django enables the development of powerful web applications that leverage predictive analytics and intelligent decision-making capabilities. Django, a high-level Python web framework, provides a robust environment for building and managing web applications, while machine learning models can enhance these applications with advanced data processing and prediction features. By combining Django's ease of use with machine learning's predictive power, developers can create dynamic and responsive systems that can analyze data, make predictions, and automate tasks based on learned patterns. This integration typically involves training machine learning models using libraries such as Scikit-Learn, TensorFlow, or PyTorch, and deploying them within Django's architecture to serve real-time predictions through web interfaces.

1. Model Development: Develop and train machine learning models using Python libraries. This involves data collection, preprocessing, model selection, training, and evaluation. The trained model is then serialized (saved) using tools like pickle or joblib for later use.

2. Django Integration:

To Create a Django Project: Set up a new Django project and application. Configure the project settings and database connections.

Develop Views and URLs: Create Django views to handle requests, process input data, and call the machine learning model for predictions. Define URLs to route requests to the appropriate views.

Load the Model: In the Django views, load the serialized machine learning model and use it to make predictions based on user input.

Handle Data: Implement forms or API endpoints in Django to capture user input, preprocess it as needed, and pass it to the machine learning model.

Display Results: Render the prediction results on web pages or provide them through APIs.

3. **Deployment:** Deploy the Django application with integrated machine learning models to a production server. Ensure that the server environment supports Python and the required machine learning libraries.

Use Cases:

· **Recommendation Systems:** Provide personalized recommendations based on user behaviour or preferences.

· **Fraud Detection:** Analyse transactions in real-time to detect and prevent fraudulent activities.

· **Chatbots:** Enhance chatbot functionality with natural language processing and predictive capabilities.

3.2. ML Model Building

Building a machine learning model involves a systematic process that starts with clearly defining the problem and translating it into a specific task, such as classification or regression. This is followed by gathering relevant data from various sources while ensuring privacy and ethical considerations. Data is then preprocessed by cleaning, transforming, and splitting it into training, validation, and test sets. Choosing the appropriate algorithm and tools is crucial for model training, which involves fitting the model to the data, tuning hyperparameters, and validating performance. The model is then evaluated using metrics and cross-validation, tested on unseen data, and deployed into a production environment. Continuous monitoring, documentation, and maintenance are essential to ensure ongoing accuracy, with feedback used to iteratively improve the model over time.

3.3 Existing Algorithm

Decision Tree: A Decision Tree is a fundamental machine learning algorithm used for solving classification and regression problems. It represents decisions and their potential consequences through a tree-like structure. Each internal node in the tree denotes a feature or attribute, each branch signifies a decision rule or outcome, and each leaf node indicates a final class label or continuous value. This graphical representation helps visualize how decisions are made and how different attributes contribute to the prediction.

How decision tree work

Decision Trees operate through a recursive process known as recursive partitioning. The algorithm starts at the root node, which contains the entire dataset. It selects a feature that best separates the data according to a chosen criterion, such as Information Gain (based on entropy) for classification or Mean Squared Error (MSE) for regression. This feature is used to split the data into subsets, creating branches for each possible outcome. This process is repeated for each subset, creating new nodes and branches until one of the stopping conditions is met, such as a maximum depth or a minimum number of samples in a node. The result is a tree where each path from the root to a leaf represents a sequence of decisions that lead to a final prediction. Decision Trees are easy to understand and interpret, as they represent decisions in a straightforward, hierarchical manner. In a project focused on detecting phishing URLs, Decision Trees can be particularly useful due to their ability to handle both numerical and categorical features effectively. They can be trained to distinguish between legitimate and malicious URLs based on various attributes like domain names, URL lengths, the presence of special characters, and the structure of the URL. By learning from historical data of phishing and non-phishing URLs, the Decision Tree can identify patterns and rules that are indicative of phishing attempts. This makes it possible to classify new URLs as either benign or malicious in a straightforward and interpretable manner. Additionally, Decision Trees provide a clear explanation of the decision-making process, which is valuable for understanding why a particular URL was classified in a certain way.

Disadvantages:

Despite their advantages, Decision Trees have notable limitations. They are prone to overfitting, particularly when the tree becomes very deep and complex. Overfitting occurs when the model learns noise and specific details from the training data that do not generalize well to new, unseen data. This results in a model that performs well on training data but poorly on validation or test data. Decision Trees can also be unstable; small changes in the training data can lead to significant changes in the tree structure, affecting the model's reliability. Furthermore, they can be biased towards features with more levels, which may not always be relevant for the classification task. Pruning techniques, which involve cutting back the tree to remove nodes that provide little additional power, and ensemble methods such as Random Forests, which combine multiple Decision Trees, are often used to address these issues and improve the model's performance and stability.

3.4 Proposed Algorithm

Random Forest

Random Forest is an ensemble learning method that combines multiple Decision Trees to improve the performance and robustness of predictions. It constructs a "forest" of Decision Trees by training each tree on a random subset of the data and features, then aggregates the predictions of all the trees to make a final decision. This approach leverages the strengths of Decision Trees while mitigating their weaknesses, resulting in a more accurate and generalizable model. The Random Forest algorithm involves creating a large number of Decision Trees during training and using techniques like bagging (bootstrap aggregating) and feature randomness to ensure diversity among the trees.

Architecture:

Random Forest operates through a two-step process: bagging and feature randomness. During the training phase, it generates multiple subsets of the original dataset by sampling with replacement (bootstrapping). Each Decision Tree is trained on a different subset, ensuring that each tree is exposed to different portions of the data. Additionally, when splitting nodes, Random

ISSN NO: 9726-001X
Volume 13 Issue 02 2025

Forest selects a random subset of features rather than considering all features, which further promotes diversity among the trees. Once trained, the Random Forest aggregates the predictions from all individual trees, typically using majority voting for classification or averaging for regression. This aggregation helps to smooth out the predictions and reduce the risk of overfitting that is common in single Decision Trees.

In the context of detecting phishing URLs, Random Forest can provide significant advantages over individual Decision Trees. By combining multiple Decision Trees, Random Forest improves the model's ability to generalize to new, unseen data and reduces the likelihood of overfitting. It can handle a large number of features and interactions between them more effectively, which is crucial for distinguishing between legitimate and malicious URLs based on complex patterns. The ensemble nature of Random Forest means that it can capture a broader range of decision rules and relationships, leading to more accurate and reliable phishing detection. Additionally, the feature importance scores provided by Random Forest can help identify the most significant attributes for classification, aiding in feature selection and model interpretability.

Disadvantages:

· **High Accuracy**: Random Forest typically provides high accuracy due to its ensemble approach. By aggregating the predictions of multiple Decision Trees, it reduces the risk of overfitting and variance, leading to more accurate predictions.

· **Robustness**: It is robust against overfitting, especially when dealing with noisy data. The use of multiple trees helps smooth out the influence of outliers and anomalies.

· **Handles Large Datasets**: Random Forest can handle large datasets with a large number of features effectively. It can manage high-dimensional data and capture complex interactions between features.

· **Feature Importance**: It provides insights into the importance of different features in the prediction process. This can be useful for feature selection and understanding which attributes are most influential.

· **Versatility**: Random Forest can be used for both classification and regression tasks. Its flexibility makes it applicable to a wide range of problems.

· **Reduced Variance**: By averaging the predictions of multiple trees, Random Forest reduces the variance of the model, which enhances its generalization to new data.

· **No Need for Feature Scaling**: Random Forest does not require feature scaling or normalization, which simplifies the preprocessing of data.

· **Handles Missing Values**: It can handle missing values in the dataset, as it can use surrogate splits to handle missing values during the decision-making process.

· **Low Risk of Overfitting**: Due to the random sampling of data and features, Random Forest is less likely to overfit the training data compared to a single Decision Tree.

· **Parallel Processing**: The training of multiple Decision Trees can be done in parallel, which can speed up the process and make it suitable for large-scale problems.

Comparison: Random Forest vs. Decision Tree

Random Forest is generally considered superior to individual Decision Trees in many scenarios, including phishing detection. This is due to its ability to reduce overfitting, improve accuracy, and handle a large number of features and complex patterns more effectively. While Decision Trees are simple and interpretable, they can be prone to overfitting and instability. Random Forest addresses these issues by averaging the predictions of multiple trees, which enhances model robustness and generalization. Overall, Random Forest's ensemble approach offers a more reliable and powerful solution for detecting phishing attempts compared to single Decision Trees.

IV. RESULTS & DISCUSSIONS

IMPLEMENTATION AND DESCRIPTION

4.1. Data Preparation

- **Import Libraries**: Start by importing necessary libraries like pandas, NumPy, scikit-learn, and others for data manipulation, model building, and evaluation.
- **Load the Dataset**: Load your dataset using pandas. The data might need to be split into features (X) and the target variable (y).
- **Data Cleaning**: Handle missing values, if any, by filling them in or dropping rows/columns. Perform any necessary transformations, such as encoding categorical variables.
- **Feature Selection**: Select the relevant features that will be used for training the model. This can be done manually or using automated techniques.
- **Train-Test Split**: Split the dataset into training and testing sets. A common split is 80% for training and 20% for testing.

4.2. Model Building

- **Initialize the Random Forest Classifier**: Create an instance of the RandomForestClassifier from scikit-learn. You can specify parameters like the number of trees (n_estimators), depth of the trees (max_depth), and others.
- **Train the Model**: Fit the model to the training data. The model will create multiple decision trees using random subsets of the data and features.
- **Feature Importance**: After training, you can extract feature importance scores to understand which features contribute the most to the predictions.

4.3. Model Evaluation

- **Predictions on Test Data**: Use the trained Random Forest model to make predictions on the test data.
- **Performance Metrics**: Evaluate the model using metrics such as accuracy, precision, recall, F1 score, and confusion matrix. These metrics help to understand how well the model is performing.
- **Cross-Validation**: To ensure that the model generalizes

ISSN NO: 9726-001X
Volume 13 Issue 02 2025

well, you can perform cross-validation, where the model is trained and validated on different subsets of the data.

4.4. Tuning the Model

- **Hyperparameter Tuning**: Use techniques like Grid Search or Random Search to find the best set of hyperparameters for the model. This can involve adjusting the number of trees, the depth of trees, and other parameters.
- **Re-training**: Train the model again with the optimized parameters to achieve better performance.

4.5. Prediction and Output

- **Final Predictions**: Use the trained model to make predictions on new, unseen data.
- **Output**: The predictions can be outputted as probabilities or classes, depending on the problem. You can save the model for future use or integrate it into a larger application.

4.6. Deployment (Optional)

- **Model Serialization**: Save the trained model using tools like joblib or pickle for deployment in a production environment.
- **Integration**: Integrate the model into a web application or API for real-time predictions.

Description of Code:

- The code begins by importing necessary libraries like pandas for data handling and RandomForestClassifier from scikit-learn for building the model.
- The dataset is loaded, cleaned, and split into features and target variables. Any necessary transformations or encoding is done at this stage.
- The dataset is then split into training and testing sets to ensure the model is evaluated on unseen data.
- A RandomForestClassifier is instantiated with specific hyperparameters, and the model is trained on the training data.
- After training, predictions are made on the test data, and various performance metrics are calculated to assess the model's accuracy and effectiveness.
- The model might be fine-tuned using hyperparameter optimization techniques to improve performance, followed by re-training with the best parameters.
- Finally, the model can be saved and integrated into an application for real-time predictions or further analysis.

Results and Discussion

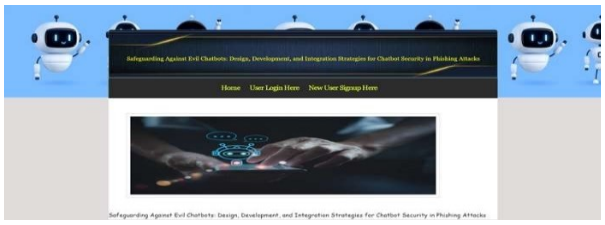

Safeguarding Against Evil Chatbots: Design, Development, and Integration Strategies for Chatbot Security in Phishing Attacks
Home
User Login Here
New User Signup Here
Safeguarding Against Evil Chatbots: Design, Development, and Integration Strategies for Chatbot Security in Phishing Attacks

Fig 2: Home Page

The homepage of the application features a clean and user-friendly design that welcomes visitors with a combination of visual and textual elements. At the top of the page, a navigation bar provides quick access to essential sections, including links to Home, User Login, and Sign Up. This allows users to easily navigate the site, whether they are returning users looking to log in or new visitors wanting to create an account. Below the navigation bar, a prominent image captures attention and sets the tone for the site's theme. Accompanying the image is a content section that offers an introductory overview, providing visitors with insights into the website's purpose, features, or services. This layout ensures a balanced presentation of information and aesthetics, making the homepage both informative and visually appealing.

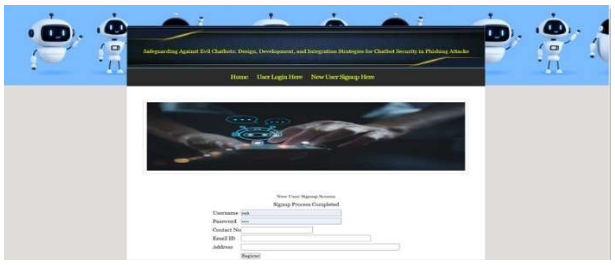


Fig 3: Signup page

The Signup Action function handles the user registration process in the application. When a user submits the signup form, the function receives the data via a POST request. It extracts the user's input, including the username, password, contact information, email, and address. The function first checks if the username already exists in the database to prevent duplicate accounts. If the username is available, the function inserts the new user's details into the register table, completing the signup process. Upon successful registration, a status message is generated to inform the user of the outcome. The function then renders the Register.html template, passing the status message to the context to provide feedback to the user. This process ensures that each user is uniquely registered and receives confirmation of their registration status.

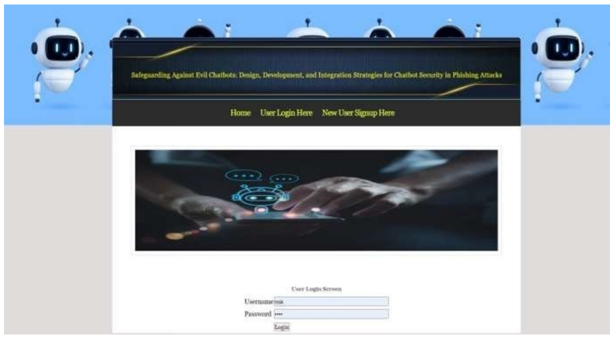

Safeguarding Against Evil Chatbots: Design, Development, and Integration Strategies for Chatbot Security in Phishing Attacks
Home
User Login Here
New User Signup Here
User Login Screen
Username
Password
Login

Fig 4: Login page

The User Login Action function manages the user login process. When a user submits their login credentials through a POST request, the function retrieves the username and password from the form data. It then connects to the database and fetches all records from the register table. The function iterates over the records to check if the provided username and password match any existing account. If a match is found, the global variable uname is set to the username, and the user is redirected to the UserScreen.html page with a personalized welcome message. If the credentials are invalid, the user is redirected back to the
UserLogin.html page with an error message indicating an invalid

ISSN NO: 9726-001X
Volume 13 Issue 02 2025

login attempt. This function ensures that only authenticated users can access the user screen, while unauthorized attempts are handled appropriately.

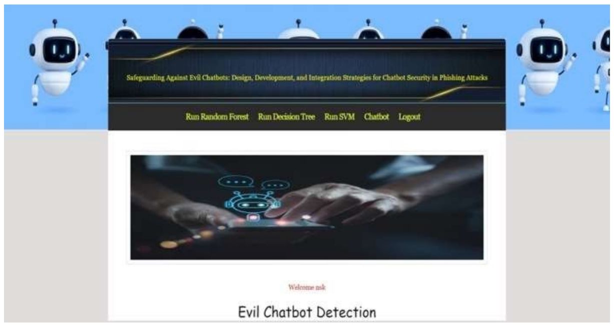

Safeguarding Against Evil Chatbots: Design, Development, and Integration Strategies for Chatbot Security in Phishing Attacks
Run Random Forest
Run Decision Tree
Run SVM
Chatbot
Logout
Welcome nsk
Evil Chatbot Detection

Fig 5: Home Page After login

The homepage after login provides users with a personalized experience, starting with a welcome message that greets them by name, creating a friendly and engaging atmosphere. The page also features a navigation bar that allows users to easily access various functionalities of the application. The navigation options include links to key features such as Random Forest, Decision Tree, and SVM, which provide access to different machine learning models. Additionally, there is a Chatbot feature that users can interact with for assistance or further engagement. Finally, the Logout option is available for users to securely exit their session when they are done. This design ensures that users have quick and easy access to important tools and features right from the homepage, enhancing their overall experience.

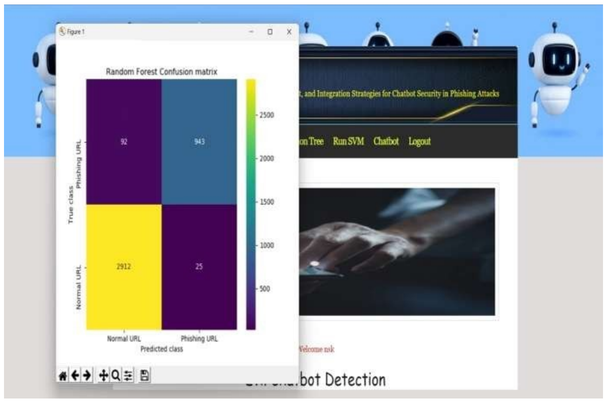
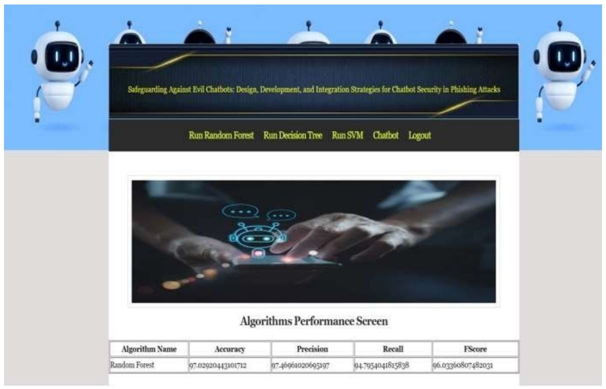


Fig 6: Run Random Forest

This section of the view function is responsible for evaluating the performance of the Random Forest classification model when a GET request is made. The function calculates key metrics such as accuracy, precision, recall, and F1-score using the test dataset. These metrics are stored in lists and then dynamically displayed in a table format on the ViewOutput.html template. Additionally, a confusion matrix is generated and visualized using a heatmap, providing insights into the model's classification performance. The heatmap distinguishes between the true and predicted classes, with labels such as 'Normal URL' and 'Phishing URL'. This visualization is intended to help users understand the model's effectiveness in distinguishing between different classes, aiding in the analysis and interpretation of the results.


Safeguarding Against Evil Chatbots: Design, Development, and Integration Strategies for Chatbot Security in Phishing Attacks
Run Random Forest
Run Decision Tree
Run SVM
Chatbot
Logout

Algorithms Performance Screen

Algorithm Name	Accuracy	Precision	Recall	FScore
Random Forest	97.02920443101712	97.46961020695197	94.7954041815838	96.03360807482031

Fig 7: Predicted Value

The Predict Action function is used to classify a given URL as either "Genuine URL" or "Contains PHISHING URL" using a pre-trained Random Forest model. When invoked with a URL input, the function first processes the URL by splitting it into components and extracting relevant data using the get Data function. This data is then transformed into a format suitable for the model using a TF-IDF vectorizer. The transformed data is fed into the Random Forest classifier (rf_cls) to make a prediction. Based on the classifier's output, the function determines if the URL is genuine or potentially a phishing attempt. The result is returned as a string indicating the classification outcome.

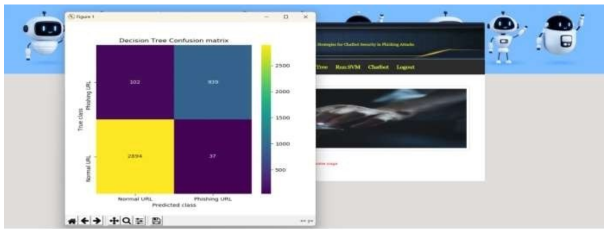


Fig 8: Decision Tree

The RunDecisionTree function evaluates the performance of the Decision Tree classification model when a GET request is made. It calculates key metrics, including accuracy, precision, recall, and F1-score, for the model using the test dataset. These metrics are appended to global lists and displayed in an HTML table on the ViewOutput.html template. The table includes performance metrics for both the Random Forest and Decision Tree models for comparison. Additionally, the function generates and displays a confusion matrix heatmap, which visualizes the model's classification results with labels for 'Normal URL' and 'Phishing URL'. This heatmap is plotted using Seaborn and Matplotlib to provide a visual representation of the model's effectiveness in distinguishing between different classes. The results are presented to the user through the rendered template, offering insights into the Decision Tree model's performance.

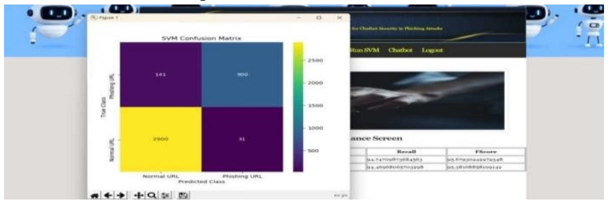


Fig 9: SVM graph

The RunSVM function is designed to assess the performance of an SVM classifier by computing key metrics such as accuracy, precision, recall, and F1-score using a test dataset. When the function is triggered by a GET request, it runs predictions on the test data (X_test) using the pre-trained SVM model (svm_cls). The computed metrics are stored in global lists and displayed in an HTML table within the ViewOutput.html template. This table also includes comparison metrics for Random Forest and Decision Tree models, allowing for easy performance evaluation across different classifiers. Additionally, the function generates a confusion matrix heatmap using Seaborn and Matplotlib, visualizing the SVM model's ability to correctly classify 'Normal URL' and 'Phishing URL' instances. The visual and tabular outputs provide comprehensive insights into the SVM model's effectiveness, which are then rendered for the user via the template.

V. CONCLUS ION

In conclusion, the increasing sophistication of cyber threats, particularly phishing attacks, necessitates the development of more advanced and adaptive security mechanisms. Chatbots, while revolutionizing customer service and user interaction, have become potential vectors for these attacks, requiring robust defenses to protect users from malicious activities. The research presented in this study focuses on the design and development of a self-defensive chatbot system, capable of detecting and neutralizing phishing attempts in real-time.

By leveraging machine learning algorithms like Support Vector Machines (SVM), Random Forest, and Decision Tree, the system achieves a high level of accuracy in distinguishing between legitimate and malicious URLs. The use of the PHISH TANK URL dataset ensures that the model is trained on a comprehensive set of real-world phishing threats, enhancing its ability to adapt to new and evolving tactics employed by attackers. The performance evaluation of the model through metrics such as accuracy, precision, recall, F-score, and confusion matrices underscores its effectiveness in providing a secure environment for chatbot interactions. This is particularly crucial in sensitive industries such as banking and finance, where the potential consequences of a successful phishing attack can be devastating, the integration of natural language processing (NLP) techniques within the chatbot not only improves the user experience but also enhances the system's ability to identify and protect sensitive information. The demonstration through a dummy banking application highlights the practical applicability of the proposed system, showcasing how it can be seamlessly integrated into existing chatbot frameworks to provide an additional layer of security. The research successfully addresses the challenges posed by phishing attacks on chatbots, offering a comprehensive solution that combines machine learning, NLP, and real-time threat detection. The self-defensive chatbot system developed in this study represents a significant advancement in chatbot security, providing organizations with a powerful tool to protect their users and maintain trust in digital interactions.

REFERENCES

[1] Yang J, Chen Y, Por L, Ku C. A systematic literature review of information security in chatbots. Appl Sci. 2023 May 23;13(11):6355. doi: 10.3390/app13116355.

[2] 18. Introducing ChatGPT. OpenAI. [2023-03-23]

[3] Sriram A, Jun H, Satheesh S, Coates A. Cold fusion: Training seq2seq models together with language models; 2017. arXiv preprint arXiv:1708.06426.

[4] Liu P, Qiu X, Huang X, Recurrent neural network for textclassification with multi-task learning; 2016. arXiv preprint arXiv:1605.05101.

[5] Serban I, Sordoni A, Bengio Y, Courville A, Pineau J. Building end-to-end dialogue systems using generative hierarchical neural network models. Paper presented at: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI. Phoenix, Arizona; 2016.

[6] Nay C. Knowing what it knows: selected nuances of Watson's strategy. IBM Res News. 2011

[7] Howard M, Lipner S. The Security Development Lifecycle. Redmond: Microsoft Press; 2006.

[8] 44. Wang D, Wang P. Two birds with one stone: two-factor authentication with security beyond conventional bound. IEEE Trans Depend Secure Comput.

[9] 2018;15(4):708-722
. https://doi.org/10:1109/TDSC.2016.2605087

[10] 45. Ometov A, Bezzateev S, Mäkitalo N, Andreev S, Mikkonen T, Koucheryavy Y. Multi-factor authentication: a survey. Cryptography. 2018;2(1):1.

[11] 46. Lemos R. Expect a new battle in cyber security: AI versus AI; 2017.

[12] 47. Joo JW, Moon SY, Singh S, Park JH. S-Detector: an enhanced security model for detecting Smishing attack for mobile computing. Telecommun Syst.

[13] 2017;66(1):29-38.

[14] 48. Milanov E. The RSA algorithm. RSA laboratories; 2009:1-11.

[15] 49. Endeley RE. End-to-end encryption in messaging services and national security–case of WhatsApp messenger. JInfSecur. 2018;9(01):95