Machine Learning for Web Vulnerability Detection

**Report**



A blue and red text on a black background

Description automatically generated

MACHINE LEARNING (INTERNS) TEAM ETA

Zarnab Zafar (Team Lead)

Muhammad Mustafa Shah

Ayesha Majeed

Wajahat Hussain

Dil Nawaz

Muhammad Abu Bakar

Ali Khan



**1.Introduction:**

As web applications continue to be a crucial part of modern digital ecosystems, the need for securing them against vulnerabilities grows in parallel. Among these vulnerabilities, **Cross-Site Request Forgery (CSRF)** is one of the most significant, allowing attackers to execute unauthorized actions on behalf of users. This can lead to data breaches, unauthorized transactions, and compromised user privacy.

Conventional vulnerability detection methods often fall short when addressing the complexity and scale of modern web applications. This project focuses on developing a machine learning (ML) solution, **Mitch**, that enhances vulnerability detection using models like Random Forest, Gradient Boosting, Support Vector Machines (SVM), and other techniques.

**2. Background:**

**2.1. Web Vulnerabilities**

Web vulnerabilities such as CSRF, SQL Injection, and Cross-Site Scripting (XSS) can have severe consequences. While traditional detection methods rely on rule-based systems and manual intervention, ML models can automate the detection process, learning from patterns in historical vulnerability data.

**2.2. Challenges in Vulnerability Detection**

Some key challenges include the dynamic nature of web applications, the growing size of attack surfaces, and the difficulty in generalizing detection mechanisms across different platforms. These challenges necessitate more intelligent and adaptive approaches, such as those provided by ML models.

**3. Project Objectives:**

This project has the following key objectives:

* **Objective 1**: Develop a machine learning system for detecting web vulnerabilities.
* **Objective 2**: Implement **Mitch**, a black-box vulnerability detection solution specifically targeting CSRF attacks.
* **Objective 3**: Compare the performance of various ML models including **Random Forest**, **Gradient Boosting**, **SVM, Naïve Bayes and others** in detecting CSRF vulnerabilities.
* **Objective 4**: Validate the effectiveness of **Mitch** through real-world data and ensure its performance through key evaluation metrics.

**4. Methodology:**

**4.1. Data Collection and Preprocessing**

Data was collected from multiple web applications, manually labeled for the presence of CSRF vulnerabilities. The data underwent preprocessing steps including:

* **Data Cleaning**: Removing irrelevant or duplicate entries to improve the quality of data.
* **Normalization**: Standardizing feature values to maintain consistency during model training.
* **Feature Selection**: Reducing noise by selecting the most important features relevant to CSRF detection.

**4.2. Model Selection**

Three machine learning models were implemented and evaluated:

* **Random Forest**: A decision-tree-based ensemble learning method known for handling large datasets and providing high accuracy.
* **Gradient Boosting**: A sequential ensemble technique that improves the accuracy of predictions by minimizing error with each iteration.
* **Support Vector Machine (SVM)**: A classification algorithm that finds the optimal hyperplane for separating different classes of data, effective for high-dimensional spaces.
* **Neural Networks**, **k-Nearest Neighbors (k-NN)**, and **Naive Bayes** were also tested to expand the scope of the analysis.

**4.3. Model Training and Tuning**

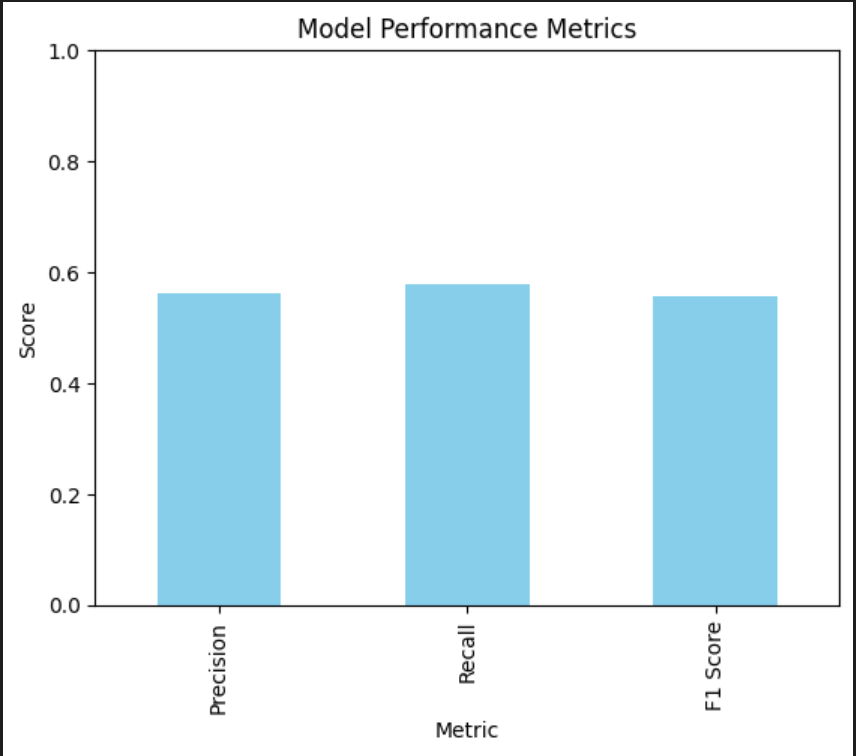
Each model was trained using the preprocessed dataset, and hyperparameters were fine-tuned to optimize their performance:

* **Random Forest**: Trained with varying numbers of trees and depth parameters to enhance performance.
* **Gradient Boosting**: Sequentially built decision trees were fine-tuned using learning rates and the number of boosting stages.
* **SVM**: Optimized using different kernel functions and regularization parameters for improved classification.
* **Neural Networks**, **k-NN**, and **Naive Bayes**: Trained and evaluated as additional models for comparison.

**5. Evaluation Metrics:**

To gauge the performance of the models, the following metrics were used:

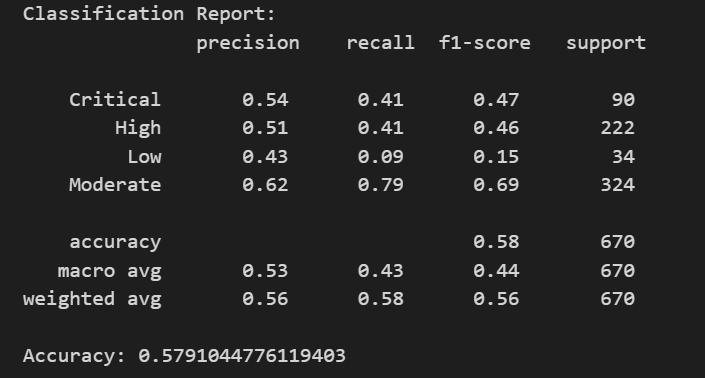
* **Accuracy**: The percentage of correct predictions out of total predictions made.
* **Precision**: The ability of the model to return only relevant instances, minimizing false positives.
* **Recall**: The capacity of the model to detect all relevant instances, focusing on minimizing false negatives.
* **F1-Score**: The harmonic mean of precision and recall, providing a balanced evaluation.

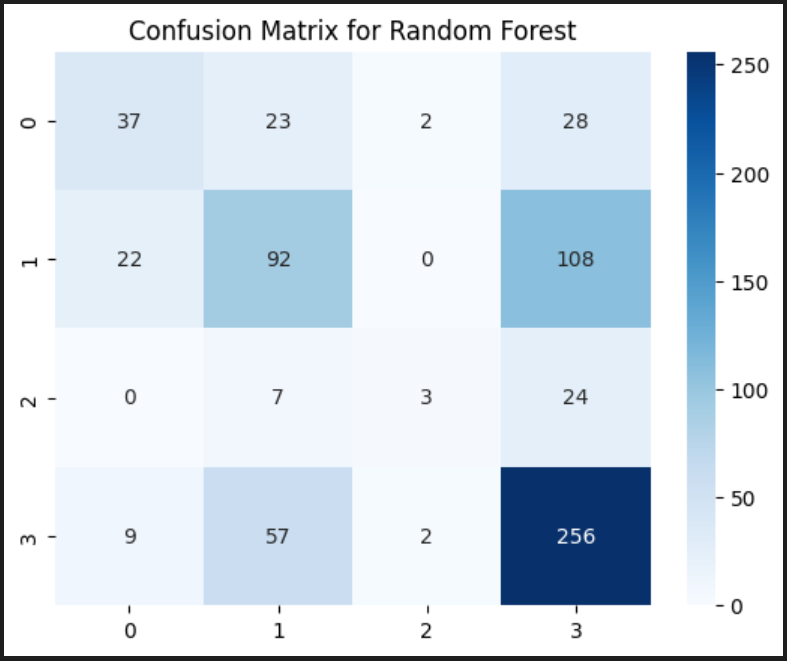


**7. Results:**

**7.1. Random Forest**

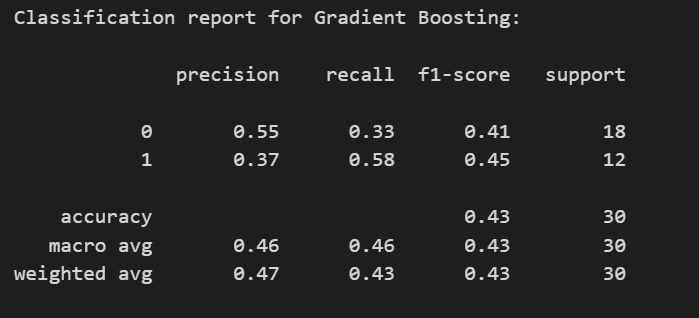
* **Accuracy**: Highest among all models.
* **Precision and Recall**: Achieved the best balance between false positives and false negatives.
* **Model Strengths**: Random Forest was the most robust and reliable model, handling the complexity of CSRF detection efficiently.





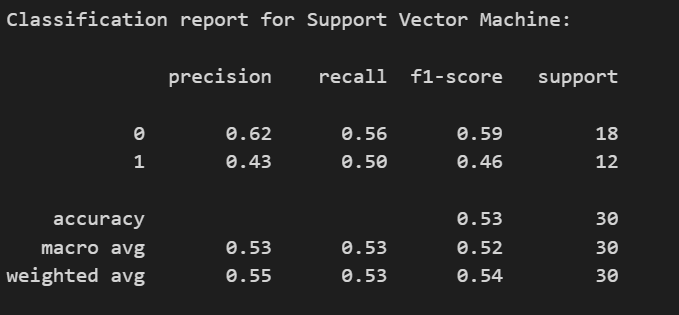
**7.2. Gradient Boosting**

* **Accuracy**: Competitive performance, slightly lower than Random Forest.
* **Precision and Recall**: Performed well in most cases, though required more computational power.
* **Model Strengths**: Gradient Boosting optimized accuracy by incrementally improving on previous mistakes, offering good results despite its computational cost.

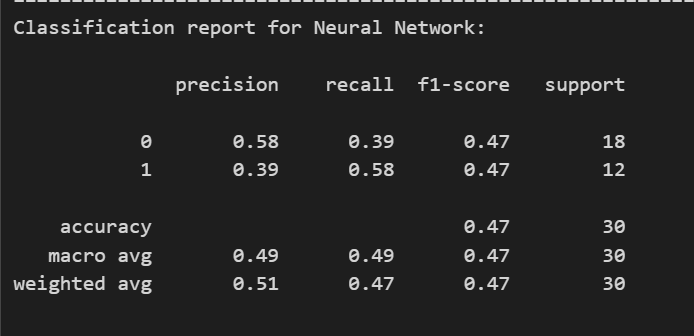


**7.3. SVM**

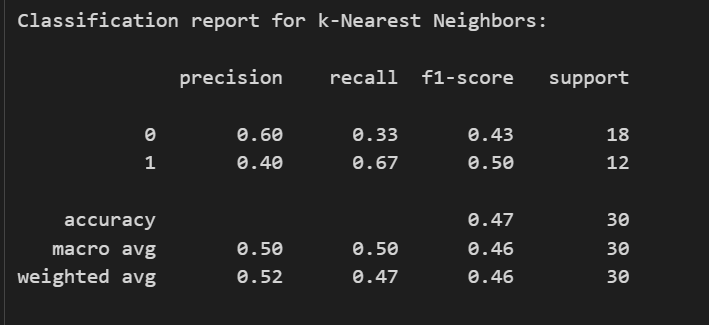
* **Accuracy**: Lower than the ensemble models.
* **Precision and Recall**: SVM struggled with generalizing across the entire dataset, producing a higher false positive rate.
* **Model Strengths**: While SVM performed reasonably well, its high computational demand made it less favorable for large-scale vulnerability detection tasks.



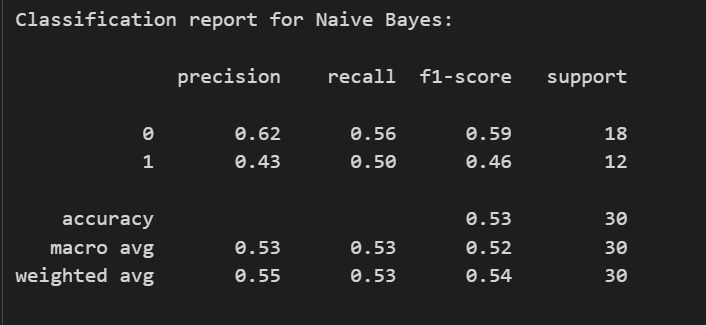
**7.4. Neural Networks**

****

**7.5. K-Nearest Neighbours:**

****

**7.5. Naive Bayes:**

****

**8. Comparison of Models:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| | **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | | --- | --- | --- | --- | --- | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Random Forest** | High | Best | Best | High | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Gradient Boosting** | High | Good | Good | High | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Support Vector Machine (SVM)** | Moderate | Lower | Moderate | Moderate | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Neural Network** | Moderate | Good | Good | Moderate | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **k-Nearest Neighbors** | Moderate | Good | Moderate | Moderate | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Naive Bayes** | Moderate | Lower | Moderate | Moderate |  | |

**9. Discussion:**

**9.1. Strengths of Random Forest**

The Random Forest model emerged as the best performer, offering the highest accuracy and the best trade-off between precision and recall. Its ensemble nature allowed it to handle the noisy and diverse dataset effectively, providing consistent results with fewer false positives and negatives.

**9.2. Computational Efficiency of Gradient Boosting**

While Gradient Boosting also performed well, its need for sequential processing made it more computationally expensive compared to Random Forest. However, it could be a viable option where accuracy needs to be maximized, despite increased computational costs.

**9.3. Limitations of SVM**

SVM, though powerful for certain types of data, struggled with this dataset, which involved high-dimensional feature spaces and imbalanced classes. Future improvements could involve using feature engineering and balancing techniques to improve SVM’s performance.

**10. Conclusion:**

The project successfully developed and evaluated an ML-based system, **Mitch**, designed to detect CSRF vulnerabilities in web applications. **Random Forest** was found to be the most reliable model, balancing high accuracy with computational efficiency. **Gradient Boosting** also performed well but required more resources, while **SVM** showed limitations with the dataset at hand.

**Mitch** marks a significant advancement in automating web vulnerability detection. Future work could involve expanding the model to detect other types of vulnerabilities like SQL Injection and XSS, further improving the security of web applications through the use of machine learning.

**Graph 5: Feature Correlation for Vulnerability Detection**

(Attach graph showing the correlation of different features for vulnerability detection)

**11. Future Work:**

In future iterations of this project, the following areas will be explored:

* **Improvement of Data Preprocessing**: More advanced preprocessing techniques like feature engineering and dimensionality reduction can enhance model performance.
* **Incorporating Additional Vulnerabilities**: While the focus was on CSRF vulnerabilities, future work will expand **Mitch** to detect a broader range of web vulnerabilities, including SQL Injection and Cross-Site Scripting.
* **Hyperparameter Tuning for SVM**: Further tuning of the SVM model could improve its classification performance, especially with larger, more complex datasets.
* **Model Deployment**: Efforts will be made to deploy **Mitch** in a real-world environment to further validate its performance and integrate it into existing web security infrastructures.