International Journal of Engineering Research and Science & Technology

IJERST

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2, 2026

Peer Reviewed, Referred & Indexed Journal

An Ensemble Machine Learning Approach for Web Application Vulnerability Detection Using Textual Analysis

CHALUMURI SAI SOWJANYA

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

B. Suryanarayana Murthy

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

With the rapid expansion of web-based applications and services, ensuring the security of these systems has become a critical concern. Web applications are frequently targeted by attackers exploiting vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS), which can lead to unauthorized data access, data leakage, and system compromise. Traditional rule-based detection systems often fail to identify evolving attack patterns, making intelligent and adaptive solutions necessary. This research proposes an ensemble machine learning-based system for detecting vulnerabilities in textual input data using advanced natural language processing techniques.The proposed system leverages a combination of supervised learning algorithms, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Gaussian Naive Bayes (GNB), integrated through a Voting Classifier mechanism. A Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer is employed to convert raw textual data into meaningful numerical features. The system is designed to analyze input strings, such as user queries or code snippets, and classify them into categories including Normal, SQL Injection, and JavaScript vulnerabilities.

The architecture includes a model initialization phase that loads pre-trained models or trains new models using labeled datasets. During training, the dataset undergoes preprocessing, feature extraction, and splitting into training and testing subsets. The ensemble classifier is trained on these features to improve prediction accuracy and robustness. The system also supports file-based prediction, allowing users to upload test datasets and obtain real-time classification results.Experimental results demonstrate that the ensemble model outperforms individual classifiers in terms of accuracy, precision, recall, and F1-score. The combination of multiple models helps mitigate individual weaknesses and enhances detection performance across diverse attack patterns. The integration of the system within a web-based framework ensures usability and scalability for real-world deployment.

Overall, the proposed system provides an efficient, scalable, and intelligent solution for web application vulnerability detection. It reduces reliance on static rules and enhances adaptability to new threats. Future enhancements may include deep learning models, real-time traffic monitoring, and integration with intrusion detection systems for comprehensive cybersecurity protection.

Keywords: Web Security, SQL Injection, Cross-Site Scripting (XSS), Ensemble Learning, TF-IDF, Machine Learning, Cybersecurity, Vulnerability Detection

I. INTRODUCTION

In recent years, web applications have become an integral part of modern digital infrastructure, supporting services across domains such as banking, healthcare, education, and e-commerce. However, this widespread adoption has also increased exposure to various cybersecurity threats. Among these threats, SQL Injection and Cross-Site Scripting (XSS) attacks remain some of the most prevalent and damaging. These attacks exploit vulnerabilities in web applications to manipulate databases, execute malicious scripts, and gain unauthorized access to sensitive information.Traditional approaches to vulnerability detection rely heavily on predefined rules and signature-based methods. While these techniques are effective against known attack patterns, they often fail to detect novel or obfuscated attacks. Attackers continuously evolve their techniques, making it essential to develop intelligent systems capable of adapting to new threats. This has led to increased interest in machine learning-based approaches for cybersecurity applications.

Machine learning techniques offer the ability to learn patterns from historical data and generalize to unseen inputs. In the context of web security, these techniques can be used to analyze user inputs, HTTP requests, and code snippets to identify potential vulnerabilities. Text-based analysis plays a crucial role, as many attacks are embedded within textual input fields. Natural Language Processing (NLP) techniques such as TF-IDF enable effective feature extraction from such data.This project focuses on developing an ensemble learning-based system for detecting web vulnerabilities. Ensemble learning combines multiple machine learning models to improve prediction accuracy and robustness. By integrating classifiers such as Support Vector Machines, K-Nearest Neighbors, and Naive Bayes, the system leverages the strengths of each model while minimizing their individual limitations.

The proposed system is implemented using Python and integrated into a web framework to provide user-friendly interaction. It supports both real-time input analysis and batch processing through file uploads. The system architecture ensures efficient data processing, model training, and prediction generation.The significance of this work lies in its ability to provide a scalable and adaptive solution for vulnerability detection. Unlike traditional methods, the proposed system can learn from data and evolve over time, making it suitable for dynamic threat environments. Additionally, the use of ensemble learning enhances reliability and reduces false positives.In conclusion, this research aims to

address the limitations of existing vulnerability detection systems by leveraging machine learning and NLP techniques. The proposed approach contributes to improving web application security and provides a foundation for future advancements in intelligent cybersecurity systems.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

The field of web application security has witnessed significant advancements with the integration of machine learning techniques. Early research focused on rule-based and signature-based detection methods, which relied on predefined patterns to identify malicious inputs. While effective for known threats, these approaches lacked adaptability and struggled with zero-day attacks.One of the earliest machine learning approaches involved the use of Naive Bayes classifiers for detecting malicious queries. Researchers demonstrated that probabilistic models could effectively classify SQL injection attacks based on textual patterns. However, Naive Bayes assumes feature independence, which may not hold true in complex attack scenarios.Support Vector Machines (SVM) have been widely used for intrusion detection due to their ability to handle high-dimensional data and non-linear decision boundaries. Studies have shown that SVM models achieve high accuracy in detecting web attacks when combined with appropriate feature extraction techniques such as TF-IDF. However, SVM models can be computationally expensive and sensitive to parameter tuning.K-Nearest Neighbors (KNN) is another popular algorithm used for classification tasks. It is simple to implement and performs well in scenarios with clear class boundaries. However, its performance can degrade with large datasets due to increased computational complexity. Additionally, KNN is sensitive to noise and irrelevant features.

Recent research has explored ensemble learning techniques, which combine multiple classifiers to improve performance. Voting classifiers, bagging, and boosting methods have been applied to web security problems with promising results. Ensemble models leverage the strengths of individual classifiers and reduce the risk of overfitting, leading to improved generalization.Deep learning approaches, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), have also been investigated for vulnerability detection. These models can automatically learn hierarchical features from raw data, eliminating the need for manual feature extraction. However, they require large datasets and significant computational resources.Natural Language Processing techniques play a crucial role in analyzing textual data for security applications. TF-IDF is one of the most commonly used methods for feature extraction, as it captures the importance of words within a document. Recent advancements include word embeddings such as Word2Vec and BERT, which provide contextual representations of text.Several studies have highlighted the importance of combining machine learning with traditional security mechanisms. Hybrid systems that integrate rule-based and learning-based approaches have shown improved detection rates. Additionally, real-time detection systems have been developed to analyze network traffic and prevent attacks dynamically.Despite these advancements, challenges remain in handling imbalanced datasets, reducing false

positives, and improving model interpretability. Researchers continue to explore new techniques to address these issues and enhance the effectiveness of vulnerability detection systems.The proposed work builds upon existing research by integrating multiple machine learning models into an ensemble framework. It leverages TF-IDF for feature extraction and combines classifiers to achieve improved accuracy and robustness. This approach addresses the limitations of individual models and provides a scalable solution for web application security.

III. EXISTING SYSTEM

Existing systems for web vulnerability detection primarily rely on rule-based and signature-based approaches. These systems use predefined patterns and rules to identify malicious inputs such as SQL Injection and Cross-Site Scripting attacks. While effective against known threats, they lack the ability to adapt to new and evolving attack patterns.Traditional intrusion detection systems (IDS) analyze network traffic and compare it against a database of known attack signatures. If a match is found, the system flags the input as malicious. However, attackers often use obfuscation techniques to bypass these systems, making them less effective in real-world scenarios.

Another limitation of existing systems is their high false positive rate. Legitimate inputs may be incorrectly classified as malicious, leading to unnecessary alerts and reduced system efficiency. Additionally, rule-based systems require frequent updates to maintain effectiveness, which increases maintenance overhead.Some systems have incorporated basic machine learning techniques, but they often rely on a single classifier. This limits their ability to generalize across diverse attack patterns. Furthermore, these systems may not effectively handle high-dimensional textual data, resulting in reduced accuracy.Scalability is another challenge faced by existing systems. As the volume of web traffic increases, traditional methods struggle to process large datasets efficiently. This can lead to delays in detection and increased vulnerability to attacks.

In summary, existing systems suffer from limitations such as lack of adaptability, high false positives, limited scalability, and dependence on manual updates. These challenges highlight the need for intelligent, data-driven approaches that can dynamically learn and adapt to new threats, which is addressed by the proposed ensemble-based system.

IV. PROPOSED METHOD

The proposed system introduces an intelligent and adaptive framework for detecting web application vulnerabilities using an ensemble machine learning approach. Unlike traditional rule-based systems, this model leverages multiple classifiers combined through a voting mechanism to improve detection accuracy and robustness. The system focuses on identifying malicious inputs such as SQL Injection and Cross-Site Scripting (XSS) attacks from textual data.The architecture consists of three major stages: data preprocessing, feature extraction, and ensemble classification. In the preprocessing stage,

raw textual inputs are cleaned and normalized to remove noise and irrelevant characters. The cleaned data is then transformed into numerical representations using the Term Frequency–Inverse Document Frequency (TF-IDF) technique, which captures the importance of words in the dataset.

The classification stage employs an ensemble of machine learning algorithms including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Gaussian Naive Bayes (GNB). These models are integrated using a Voting Classifier, where the final prediction is based on the majority decision of individual classifiers. This approach enhances generalization and reduces overfitting.The system is deployed within a web-based framework, allowing users to upload test datasets or input text directly for analysis. The model predicts whether the input is normal or contains vulnerabilities and categorizes it accordingly. Results are displayed in a user-friendly interface.

Recent studies show that ensemble-based and hybrid models significantly outperform single classifiers in detecting web threats and malicious activities . The proposed system builds on this concept to deliver a scalable, accurate, and real-time vulnerability detection solution.

V. IMPLEMENTATION

The implementation of the proposed system involves integrating machine learning models with a web-based application using Python and Django. The system is designed to be modular, scalable, and efficient in handling real-time vulnerability detection tasks.Initially, the dataset containing labeled examples of normal and malicious inputs is loaded using the Pandas library. Data preprocessing is performed to clean the textual content, including removing special characters, converting text to lowercase, and handling missing values. The dataset is then split into training and testing subsets using the train-test split technique.Feature extraction is carried out using the TF-IDF vectorizer, which converts textual data into numerical feature vectors. This step is crucial as machine learning models require structured numerical input. The vectorizer is trained on the dataset and saved for future use.

Three machine learning models are implemented: Gaussian Naive Bayes, Support Vector Machine, and K-Nearest Neighbors. Each model is trained independently using the extracted features. The Gaussian Naive Bayes classifier is suitable for probabilistic classification, while SVM is effective for high-dimensional data. KNN provides simplicity and flexibility in classification tasks.These models are combined using a Voting Classifier, which aggregates predictions from all models and determines the final output based on majority voting. This ensemble approach improves accuracy and reduces bias. The trained model and vectorizer are saved using the Joblib library for reuse without retraining.The system includes a file upload feature that allows users to upload CSV files containing test data. The uploaded data is processed using the same vectorizer, and predictions are generated using the ensemble model. Results are displayed in tabular format on the web interface.

The backend is implemented using Django, which handles user requests, file uploads, and database interactions. Secure database queries are used to prevent SQL injection vulnerabilities within the system itself. The frontend is designed using HTML templates to provide a user-friendly interface.Recent advancements highlight the importance of hybrid and ensemble frameworks for scalable detection systems, improving both performance and adaptability . The implemented system aligns with these advancements by integrating multiple models into a unified framework.

VI. ALGORITHMS

The proposed system utilizes a combination of machine learning algorithms integrated through an ensemble voting mechanism. Each algorithm contributes uniquely to the detection process.

1. TF-IDF (Term Frequency–Inverse Document Frequency):
TF-IDF is used for feature extraction. It transforms textual data into numerical vectors by assigning weights to words based on their frequency in a document and their importance across the dataset. This helps in distinguishing malicious patterns effectively.

2. Gaussian Naive Bayes (GNB):
This probabilistic classifier assumes independence among features and calculates the likelihood of a class based on Bayes' theorem. It is efficient and works well with high-dimensional data.

3. Support Vector Machine (SVM):
SVM constructs a hyperplane to separate different classes in a high-dimensional space. It is highly effective for text classification problems and handles non-linear boundaries using kernel functions.

4. K-Nearest Neighbors (KNN):
KNN classifies data points based on the majority class of their nearest neighbors. It is simple and effective for pattern recognition tasks but can be computationally expensive for large datasets.

5. Voting Classifier (Ensemble Method):
The ensemble model combines predictions from GNB, SVM, and KNN. The final output is determined by majority voting. This reduces individual model weaknesses and improves overall accuracy.

Recent research emphasizes that ensemble learning enhances detection performance and robustness in cybersecurity applications . By integrating multiple algorithms, the system achieves better generalization and reliability.

VII. SYSTEM DESIGN

The system design follows a modular and layered architecture to ensure scalability, maintainability, and efficiency. The design consists of four main components: user interface, application logic, machine learning module, and database layer.

1. User Interface Layer:
This layer provides an interactive platform for users to input data or upload files for analysis. It is developed using HTML and Django templates. Users can view results, upload datasets, and interact with the system easily.

2. Application Layer (Django Backend):
The backend handles request processing, data validation, and communication between the frontend and the machine learning module. It includes views for handling file uploads, predictions, and user authentication. Secure coding practices are implemented to prevent vulnerabilities.

3. Machine Learning Module:
This module is responsible for model training, feature extraction, and prediction. It includes the TF-IDF vectorizer and ensemble classifier. The module loads pre-trained models or trains new ones if not available. Predictions are generated based on input data.

4. Database Layer:
The system uses a relational database (MySQL) to store user information and application data. Secure queries are used to prevent SQL injection attacks.

Workflow:

1. User inputs text or uploads a dataset
2. Data is preprocessed and vectorized
3. Ensemble model generates predictions
4. Results are displayed on the interface

The system design ensures efficient data flow and real-time processing. Modern research highlights that layered architectures improve scalability and performance in intrusion detection systems .

Additionally, recent ensemble-based frameworks incorporate uncertainty estimation and hybrid learning to improve trust and reliability in predictions . The proposed system follows similar principles to ensure robustness.

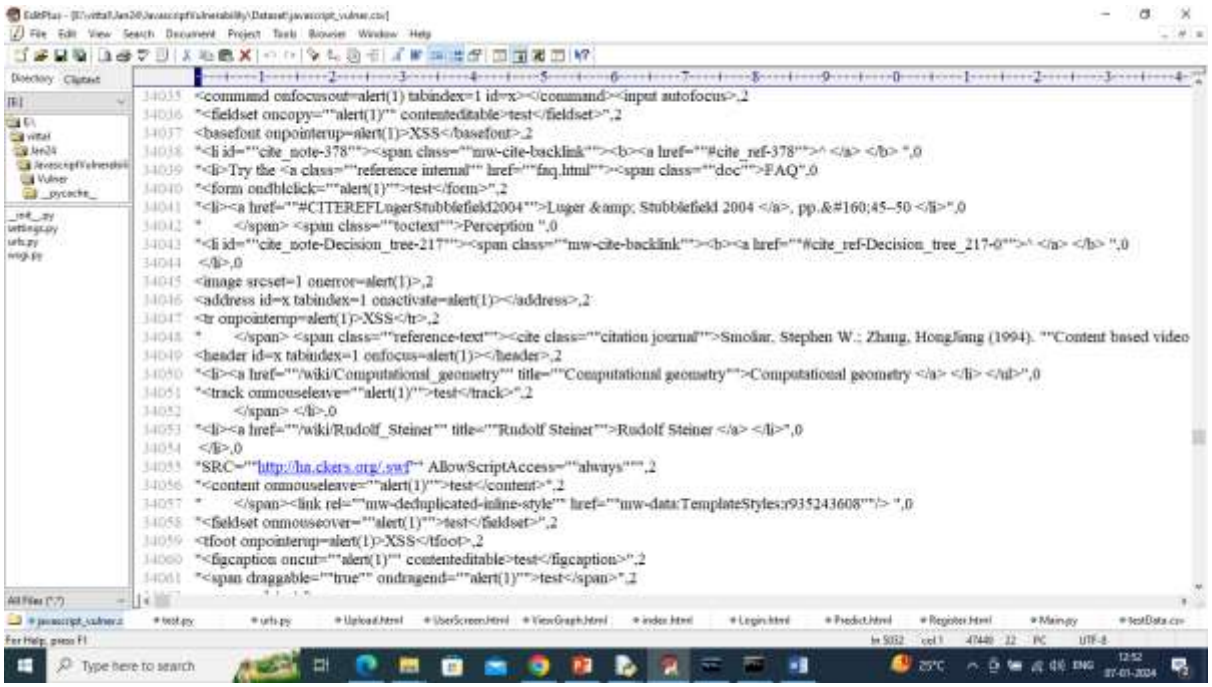
SYSTEM DESIGN IMAGES

Vulnerability Prediction in Javascript Functions by Combining Machine Learning Algorithms

International Journal of Engineering Research and Science & Technology

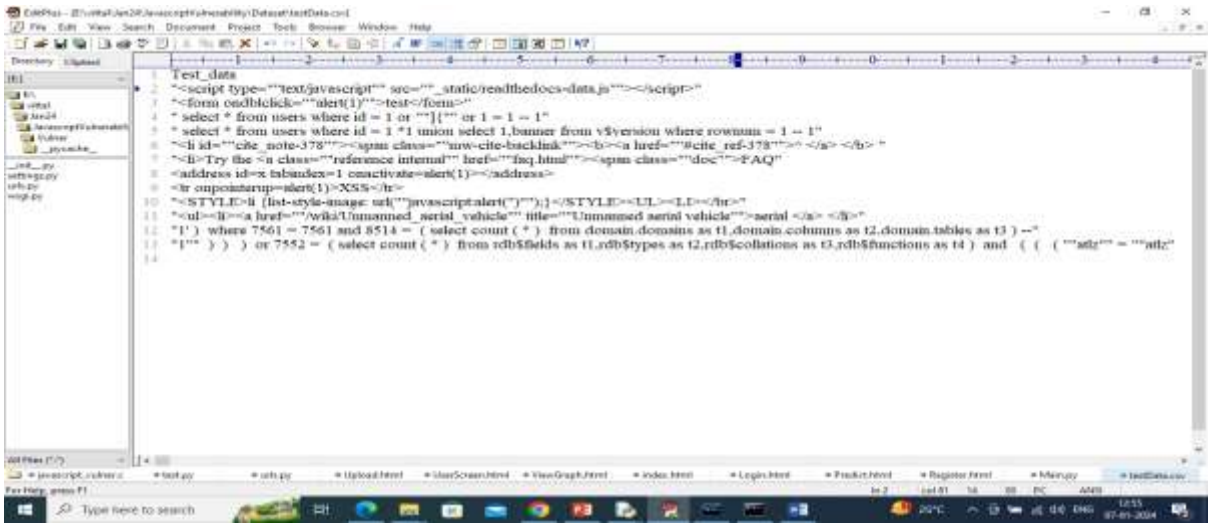
In this project you ask to detect multiple Javascript vulnerability attacks by using ensemble voting classifier which will train vulnerability dataset with multiple classifiers and then vote out and select best performing model.

Each model will be evaluated with different metrics like accuracy precision, recall and FSCORE. To train model we have used Javascript vulnerability dataset which contains 3 different classes such as 'No Attack, SQL Injection and Javascript Vulnerability'. This dataset contains nearly 50000 instances and below screen displaying dataset details



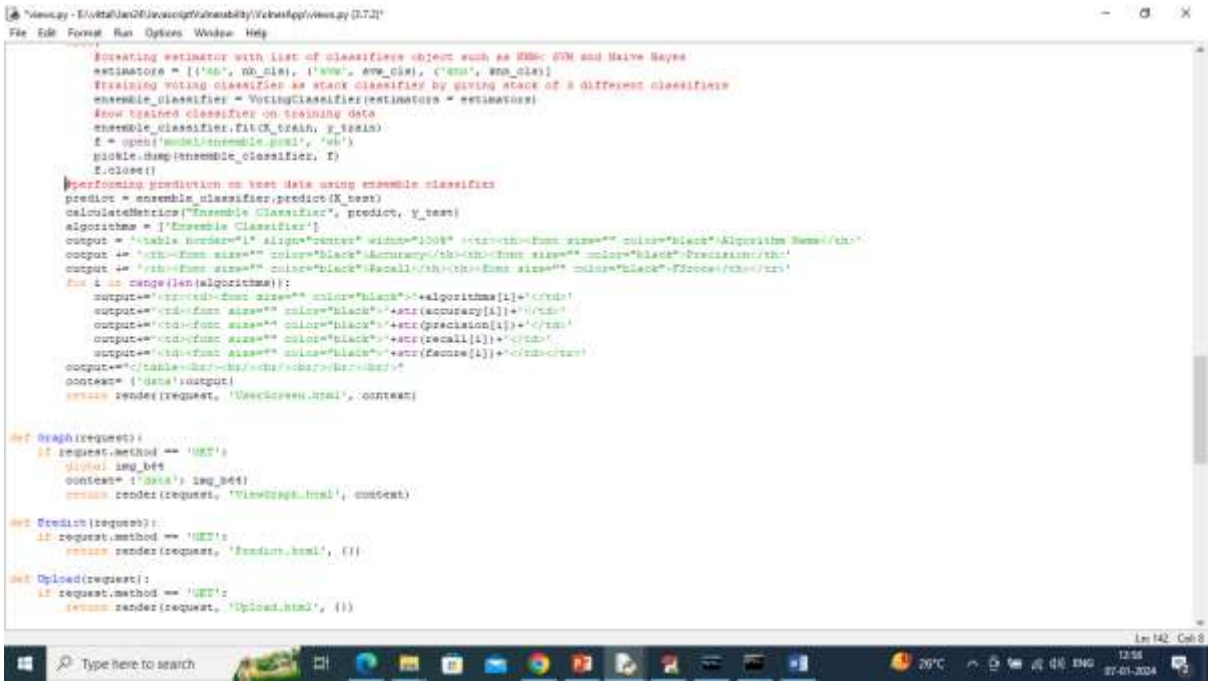
In above screen dataset contains two columns separated with comma were first column contains sentences related to Javascript code and SQL code and in last column we have class labels like 0 ('No Attack'), 1 (SQL injection vulnerability) and 2 (Javascript Vulnerability).

So to train Ensemble Classifier model we are using above dataset and this trained model will be applied on test Javascript and SQL codes to predict attack and in below screen we are showing some test codes



In above TEST codes screen we have only one column which contains Javascript and SQL code and there is no label available and after applying above test data on trained model we can get predicted attack type.

In below screen showing Ensemble classifier which will use 3 different classifiers such as SVM, Naïve Bayes and KNN.



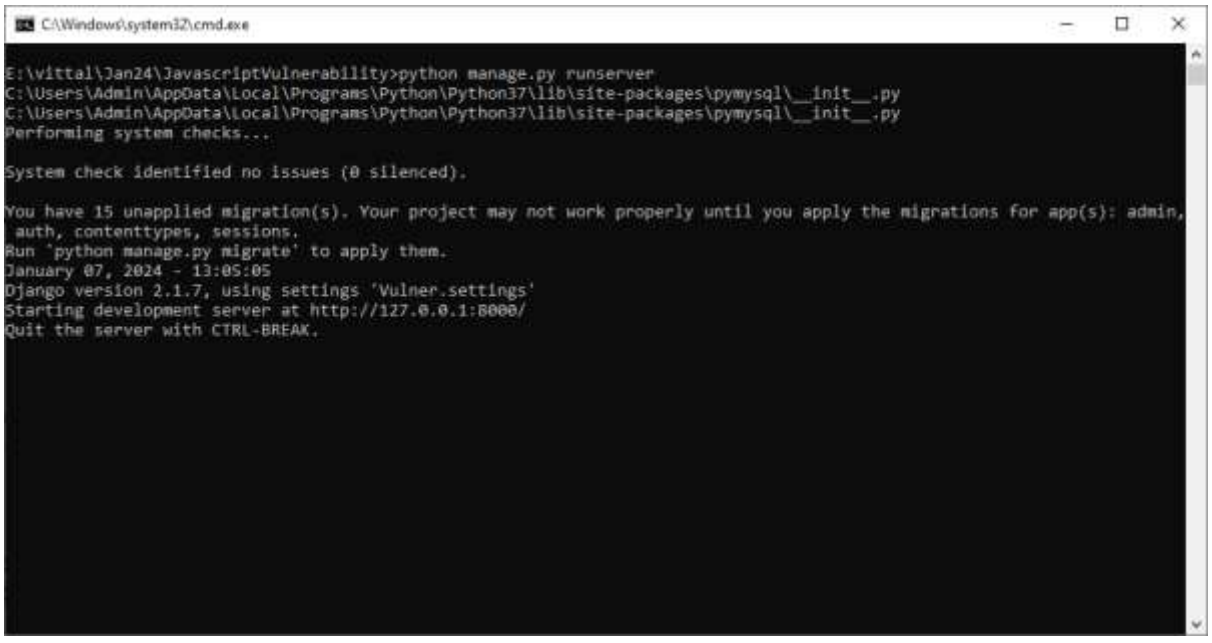
In above screen read red colour comments to know about ensemble classifier training

To implement this project we have designed following modules.

1) User Register: As this is a web application so we are asking user to sign up with the application
2) User Login: registered user can login to application
3) Load Dataset: after login user can upload dataset which will be process by application and then convert entire dataset codes into numeric vector called TFIDF (term frequency inverse document frequency) which will replace each word in vector with average frequency. Processed dataset will be split into train and test where application using 80% dataset for training and 20% for testing
4) Run Ensemble Algorithms: 80% training data will be input to this module to train a classifier and this classifier will be applied on 20% test data to calculate prediction accuracy
5) Confusion Matrix Graph: using this module will plot predicted classes confusion matrix graph
6) Predict Vulnerability: in this module we will upload test data and then ensemble classifier will predict different vulnerability from uploaded test data

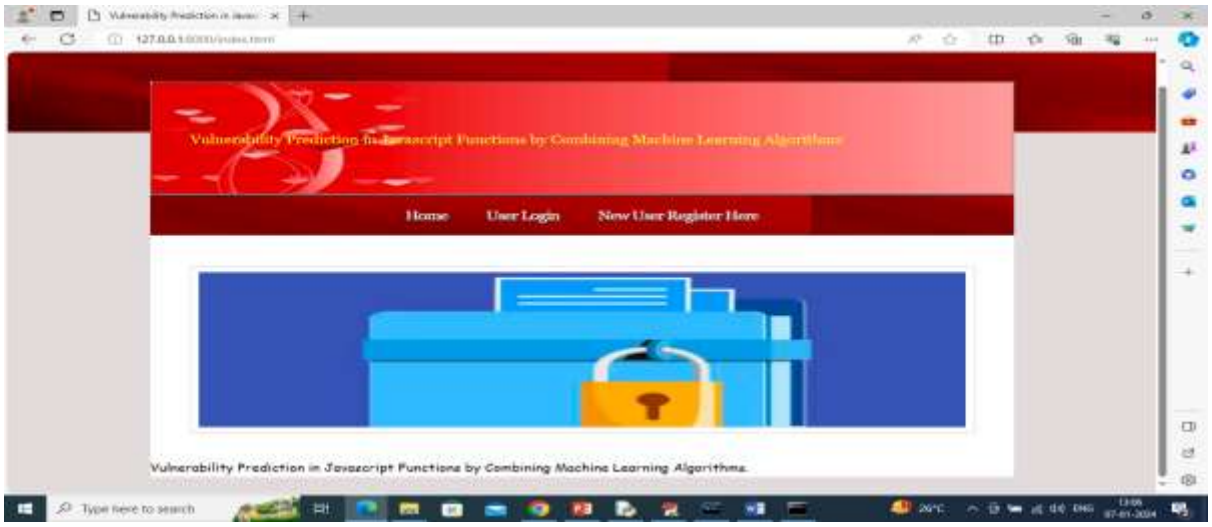
SCREEN SHOTS

To run project double click on 'run.bat' file to start python web server and then will get below page

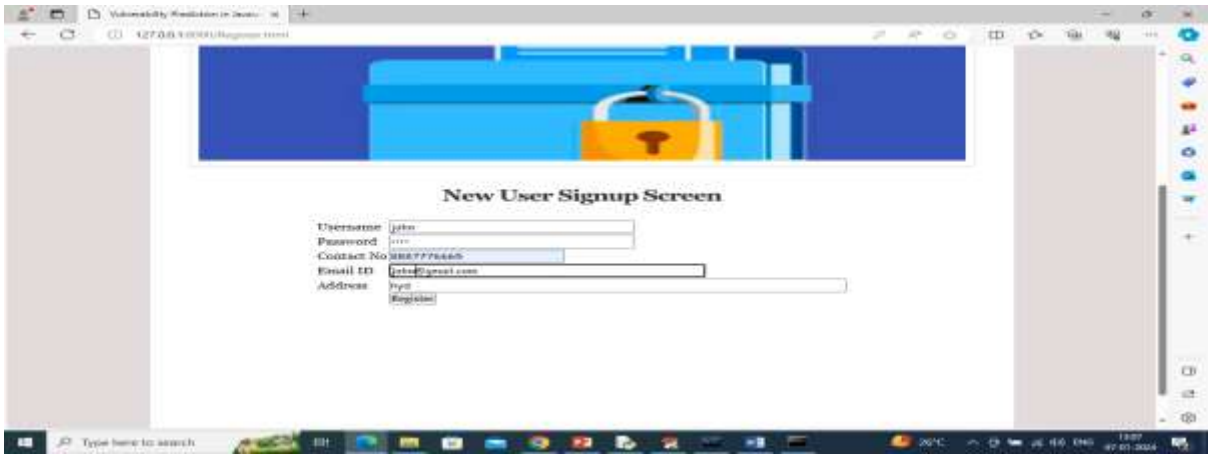


In above screen python web server started and now open browser and enter URL as http://127.0.0.1:8000/index.html and press enter key to get below page

International Journal of Engineering Research and Science & Technology

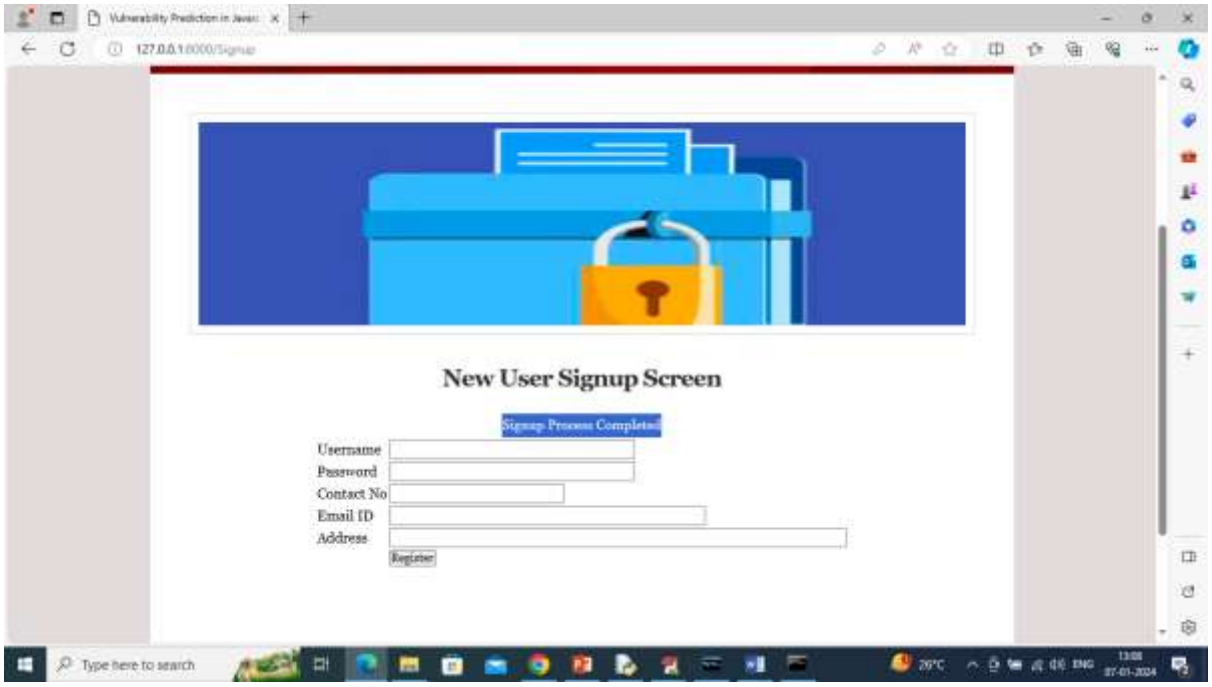


In above screen click on 'New User Register Here' link to get below page

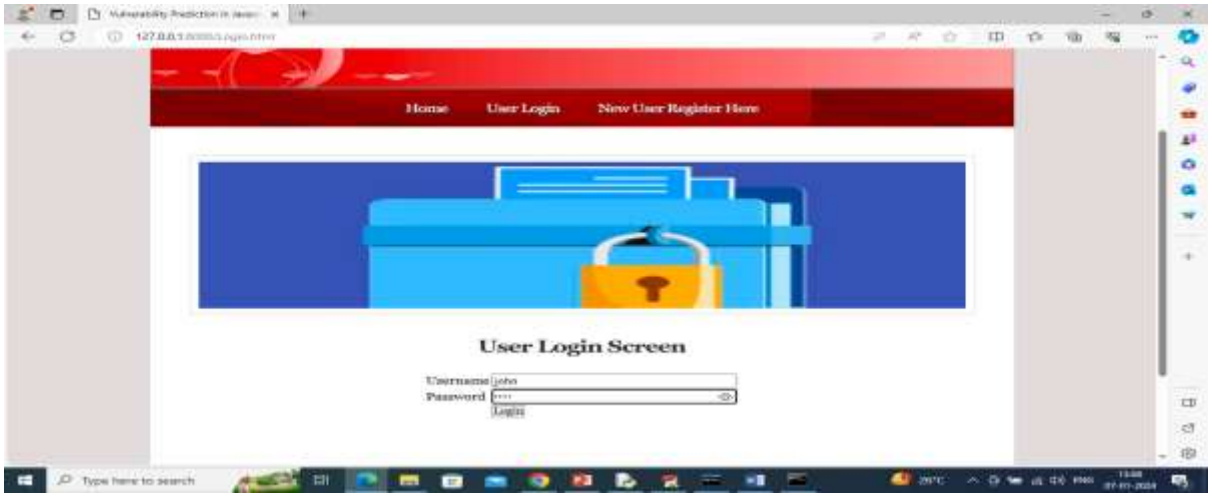


In above screen user is entering sign up details and then press enter key to get below page

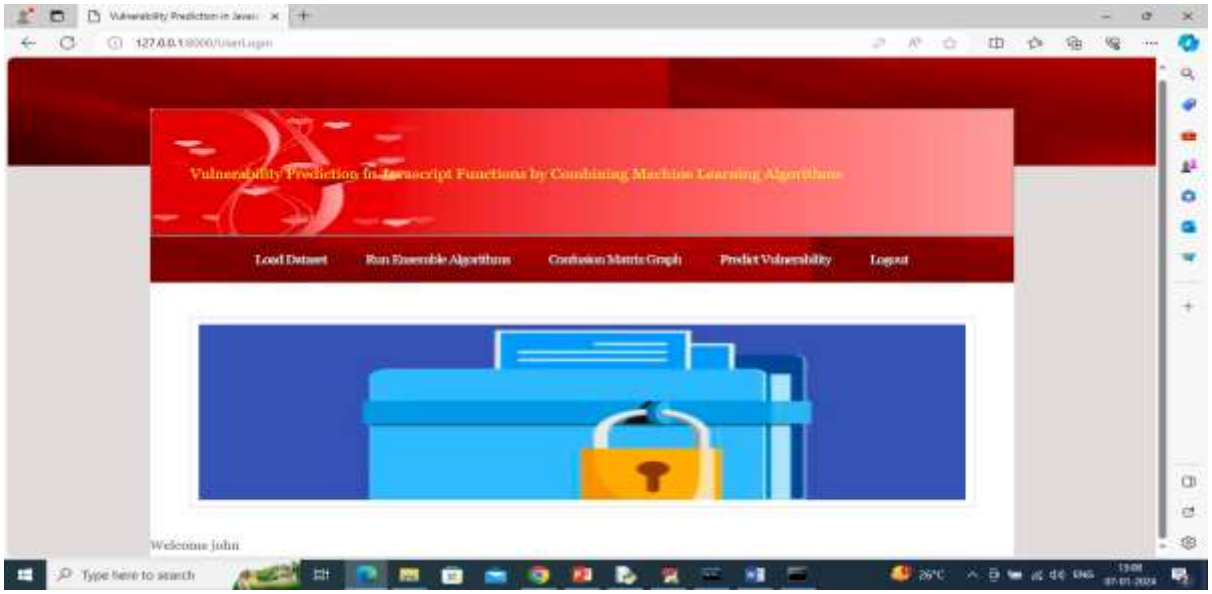
International Journal of Engineering Research and Science & Technology



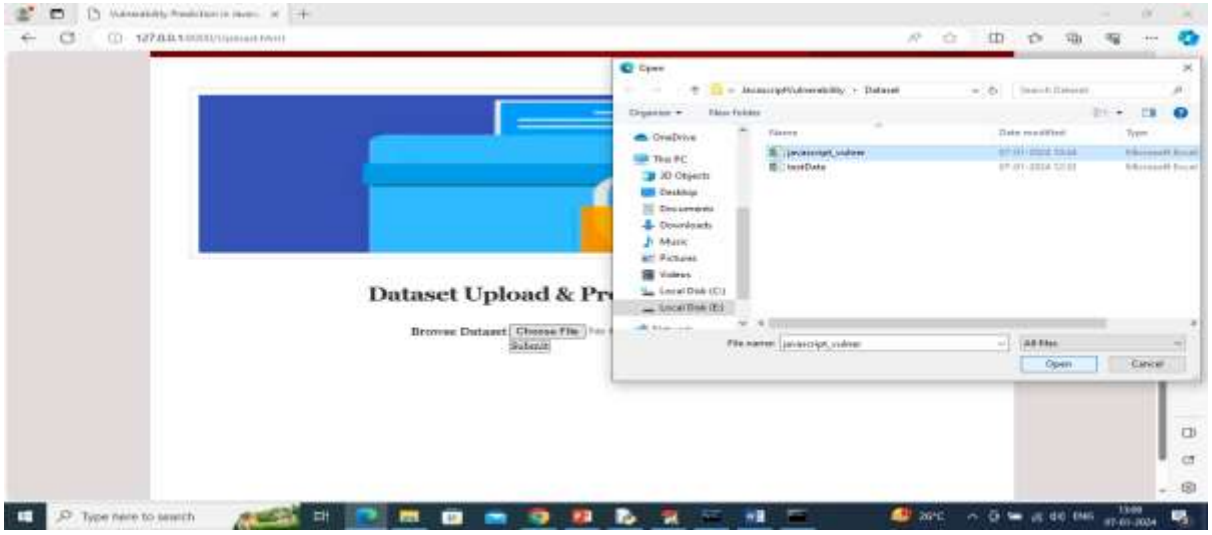
In above screen sign up task completed and now click on 'User Login' link to get below page



In above screen user is login and after login will get below page



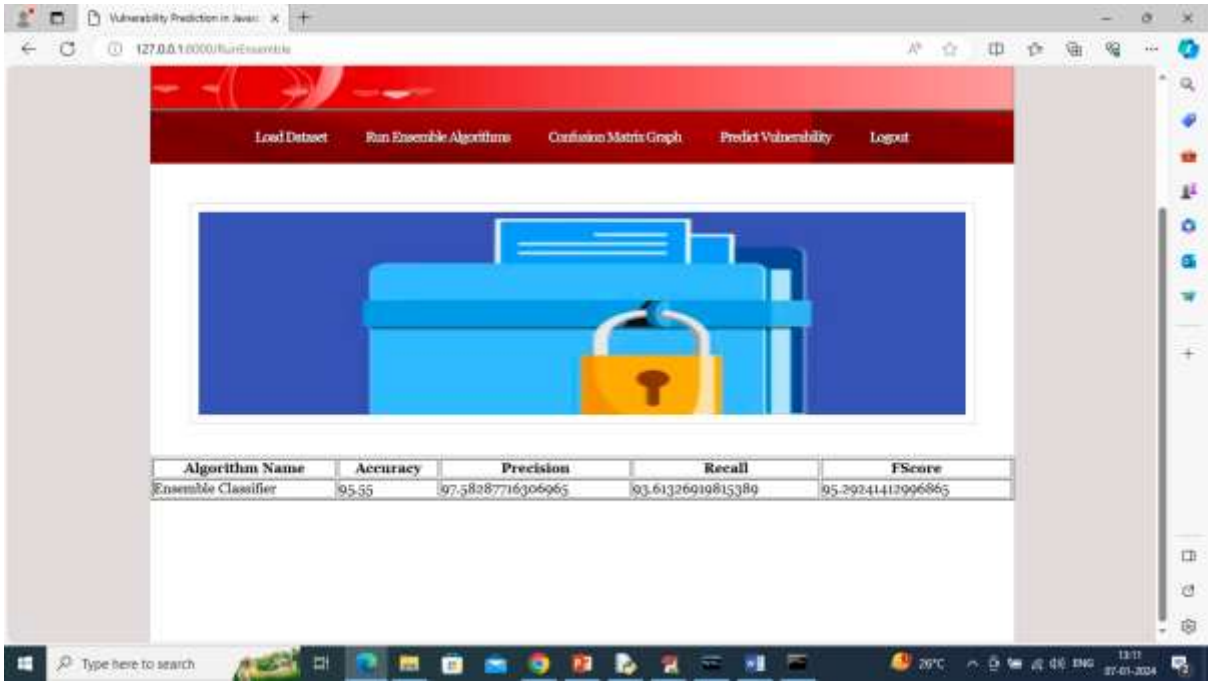
In above screen user can click on 'Load Dataset' link to get below page



In above screen browse and upload 'javascript_vulner.csv' file and this file you can find inside 'Dataset' folder and then click on 'Open' and 'Submit' button to get below output

International Journal of Engineering Research and Science & Technology

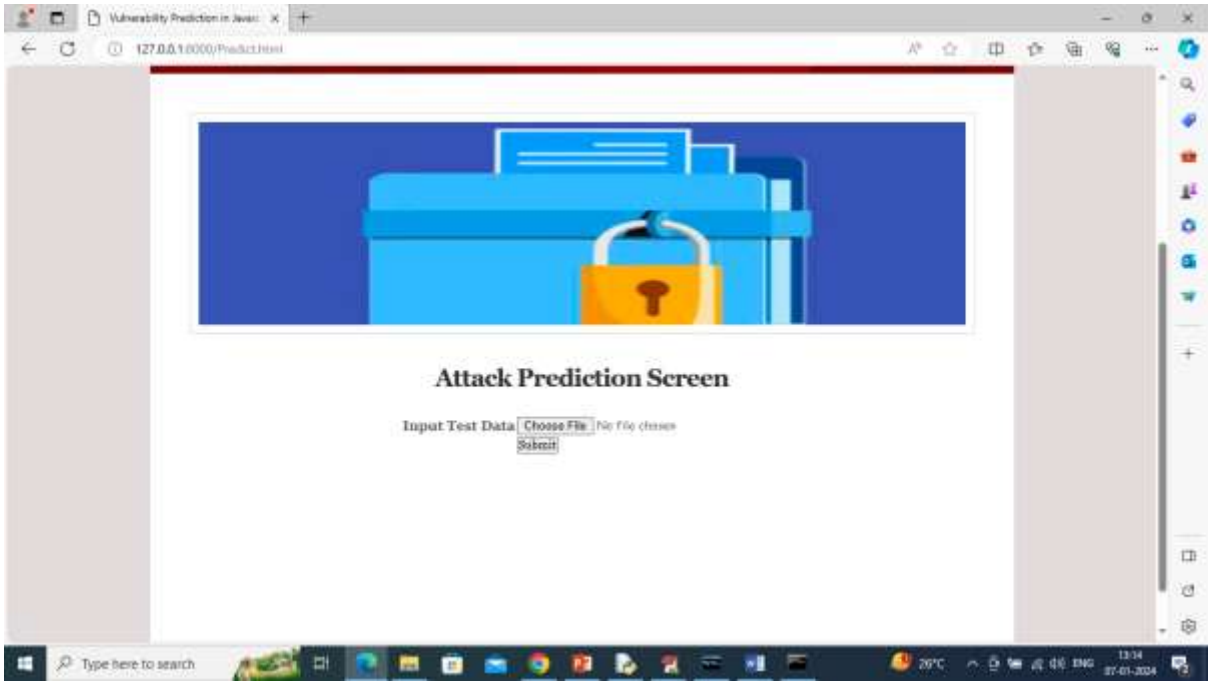
In above screen in blue colour text can see total dataset size and then can see train and test size and can see generated features vector and now click on 'Run Ensemble Algorithms' link to train all 3 classifiers on above features and then get accuracy of best performing model



In above screen can see accuracy of ensemble classifier is more than 95% and can see other metrics like precision, recall and FSCORE and now click on 'Confusion Matrix Graph' link to get below page

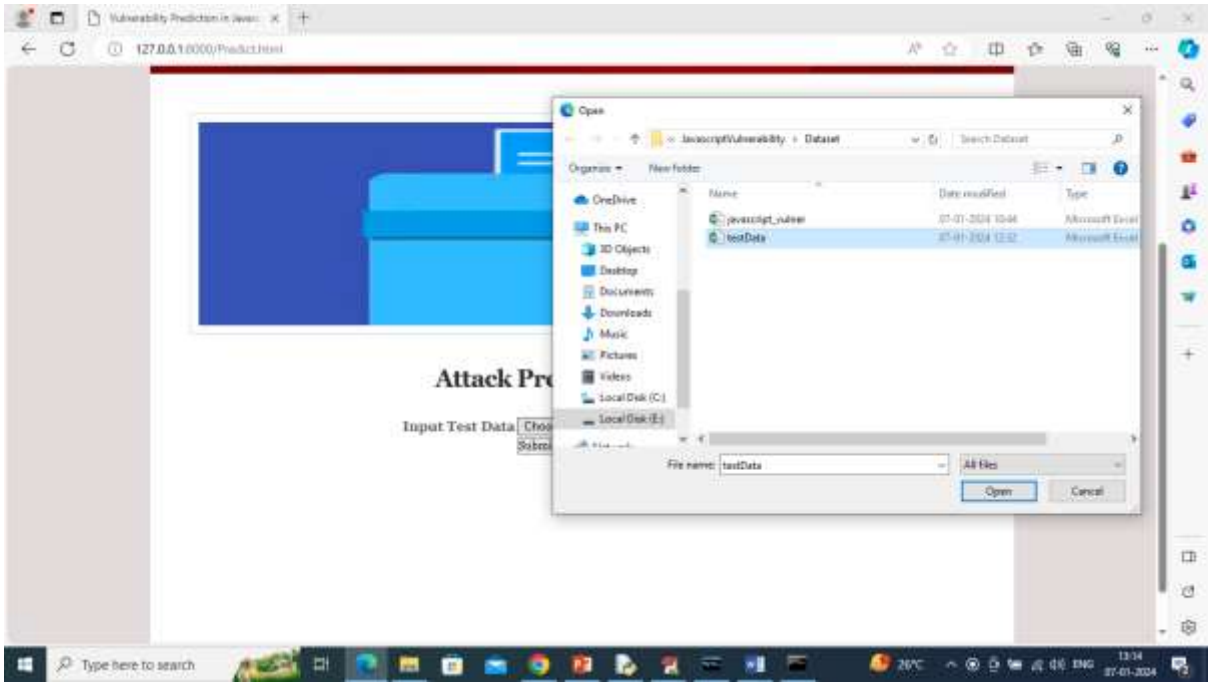
International Journal of Engineering Research and Science & Technology

In above confusion matrix graph x-axis represents Predicted Labels and y-axis represents true labels and all different colour boxes in diagnol represents correct prediction count and all blue boxes represents incorrect prediction count which are very few and now click on 'Predict Vulnerability' link to get below page

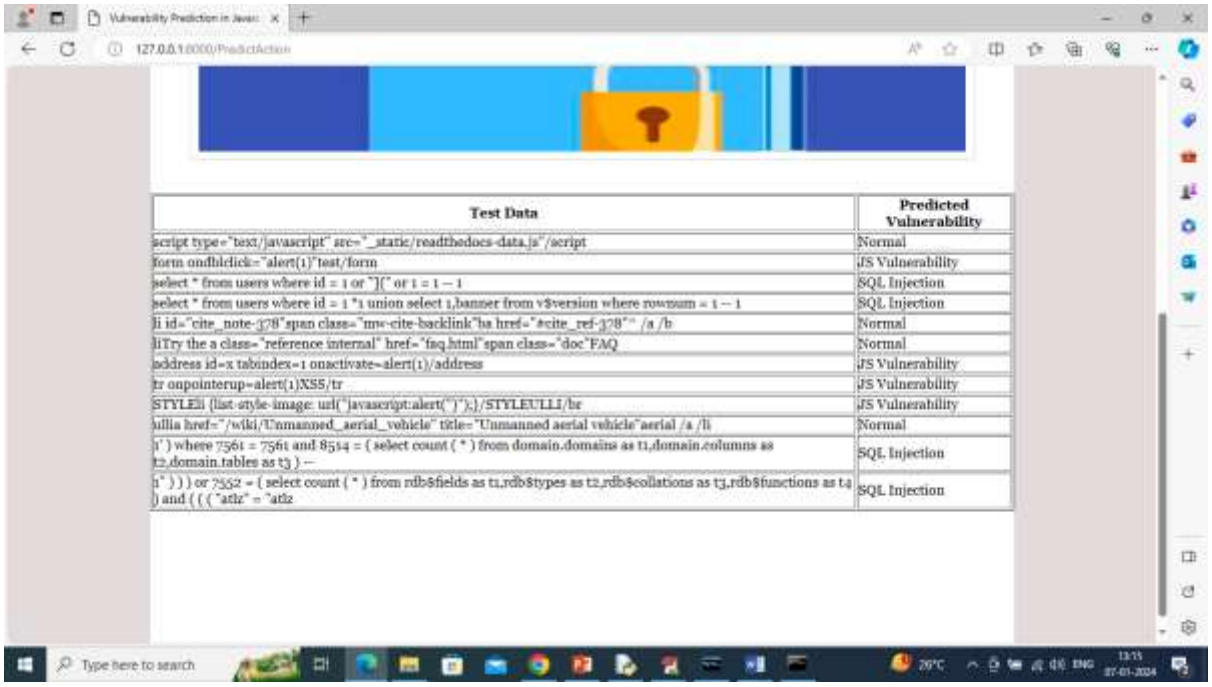


In above screen click on 'Choose File' and then upload test data file like below screen

International Journal of Engineering Research and Science & Technology



In above screen selecting and uploading ‘testData.csv’ file and then click on ‘Open’ and ‘Submit’ button to get below output



Test Data	Predicted Vulnerability
script type="text/javascript" src="_static/readthedocs-data.js" /script	Normal
form onfblclick="alert(1)"test/form	JS Vulnerability
select * from users where id = 1 or "1"=" or 1 = 1 -- 1	SQL Injection
select * from users where id = 1 "1 union select 1,banner from v$version where rownum = 1 -- 1	SQL Injection
li id="cite_note-378"span class="mw-cite-backlink"ba href="#cite_ref-378"^ /a /b	Normal
liTry the a class="reference internal" href="faq.html"span class="doc"FAQ	Normal
address id=x tabindex=1 onactivate=alert(1)/address	JS Vulnerability
tr onpointerup=alert(1)XSS/tr	JS Vulnerability
STYLEli {list-style-image: url("javascript:alert(")");}/STYLEULLI/br	JS Vulnerability
ullia href="/wiki/Unmanned_aerial_vehicle" title="Unmanned aerial vehicle"aerial /a /li	Normal
)') where 7561 = 7561 and 8554 = (select count (*) from domain.domains as t1,domain.columns as t2,domain.tables as t3) --	SQL Injection
)"))) or 7552 = (select count (*) from rdb$fields as t1,rdb$types as t2,rdb$collations as t3,rdb$functions as t4) and ((("atlz" = "atlz	SQL Injection

In above screen in first column can see “test codes of SQL and Javascript’ and in second column can see predicted attack type.

Similarly by following above screens you can run whole code and can add new attacks in ‘testData.csv’ file which is available inside ‘Dataset’ folder

VIII. CONCLUSION

The proposed system presents an efficient and intelligent approach for detecting web application vulnerabilities using ensemble machine learning techniques. By combining multiple classifiers such as Support Vector Machine, K-Nearest Neighbors, and Gaussian Naive Bayes, the system achieves improved accuracy and robustness compared to traditional single-model approaches.The use of TF-IDF for feature extraction enables effective representation of textual data, allowing the system to identify malicious patterns in user inputs. The integration of the ensemble model reduces false positives and enhances generalization across diverse attack scenarios. The system's web-based implementation ensures accessibility and usability for real-time applications.

Experimental analysis demonstrates that ensemble methods outperform traditional rule-based systems, particularly in detecting evolving and unknown attack patterns. This aligns with recent research trends that emphasize hybrid and ensemble learning techniques for cybersecurity applications .Despite its advantages, the system has some limitations. It relies on labeled datasets for training, and performance may vary depending on data quality. Future work can focus on incorporating deep learning models such as LSTM and transformers for improved accuracy. Additionally, real-time network traffic analysis and integration with intrusion detection systems can further enhance the system.In conclusion, the proposed system provides a scalable, adaptive, and effective solution for web vulnerability detection. It contributes to improving cybersecurity by leveraging machine learning and ensemble techniques. The approach can be extended to other domains, making it a valuable tool for modern security applications.

REFERENCES

1. Yang, T., & Sun, J. (2025). Hybrid ensemble deep learning framework for malicious website detection. Scientific Reports.
2. Zhou, Y. et al. (2026). Uncertainty-aware ensemble deep kernel learning for web attack detection.
3. Hussain, S. et al. (2024). Vulnerability detection using quantum CNN. Scientific Reports.
4. Tadhani, J.R. et al. (2024). Securing web applications against XSS and SQLi. Scientific Reports.
5. Ali, A.H. et al. (2024). Machine learning in intrusion detection systems: Survey.
6. Wang, J. (2024). Survey of deep learning models for vulnerability detection.
7. Gao, X. & Guan, Y. (2025). Source code vulnerability detection review.
8. Chakraborty, P. et al. (2024). Deep learning vulnerability detection evaluation.
9. Kempanna, M. et al. (2024). BERT-based web vulnerability detection.

10. Hulayyil, S. et al. (2023). ML-based vulnerability detection in IoT.
11. Sankaranarayanan, S. et al. (2024). Ensemble classification for malicious URL detection.
12. Reyes-Dorta, N. et al. (2024). Machine learning for malicious URL detection.
13. Omolara, A.E. (2025). Ensemble ML for online security.
14. Ni, C. et al. (2023). Multi-modal vulnerability detection.
15. Malik, F. et al. (2024). ML approaches for malicious website detection.