1. CHAPTER - 1
INTRODUCTION

1.1 OVERVIEW OF THE PROJECT:

Web applications are widely used and include sensitive and personal data. This makes them a target for malware that uses vulnerabilities to get unauthorized data stored on the computer. Such attacks include cross-site scripting (XSS). Cross-Site scripting (XSS) is a type of attack on web systems that involves inserting malicious code into a page issued by a web system (which will be executed on the user's computer when they open this page) and interacting this code with the attacker's web server. This is a type of "code Injection" attack. The specific feature of such attacks is that malicious code can use the user's authorization in the web system to get extended access to it or to get the user's authorization data.

Malicious code can be inserted into a page either through a vulnerability in the web server or through a vulnerability on the user's computer. The term is abbreviated "XSS" to avoid confusion with cascading style sheets that use the abbreviation "CSS". XSS can affect the victim by stealing cookies, changing the web page, capturing clipboard content, scanning ports, or dynamically loading. Therefore, the security of web applications is a very important task for developers. The lack of client input verification is the most common security flaw in web applications. These shortcomings are repeatedly detected and exploited both on the client side and on the server side. XSS attacks remain in the top ten vulnerabilities listed in the Open Web Application Security Project (OWASP). XSS attacks have not been studied for a long time.

Such attacks were considered not dangerous. However, this opinion is considered erroneous now. On a site where there is no protection against cross-site request forgery (CSRF), an attacker can perform any actions available to the user. But even with this protection, the page or HTTP Cookie may contain very vulnerable data (for example, the administrator session ID or payment document numbers). Cross-site scripting can be used to conduct a DoS attack.

Principal Component Analysis (PCA) serves as a powerful tool in data science by reducing the dimensionality of complex datasets while retaining as much variance as possible. By transforming high-dimensional data into a lower-dimensional space, PCA enables visualization and interpretation of data patterns in a more manageable form. This "projection" or "shadow" provided by PCA allows users to identify the most informative features and gain insights into the underlying structure of the data.

1.2 OBJECTIVE OF THE PROJECT:

The objective of this project is to redefine the paradigm of roadside assistance, aiming to deliver unparalleled convenience, reliability, and affordability to users while concurrently fostering community growth and development.

- **Identify XSS as a Top Web Application Risk:**

  Acknowledge XSS as one of the primary risks to web application security, as ranked by OWASP.

- **Highlight the Need for XSS Detection:**

  Emphasize the criticality of detecting XSS attacks due to their potential to compromise sensitive data and undermine user security.

- **Address Misconceptions About XSS:**

  Challenge the misconception that XSS attacks are harmless by illustrating their capacity to exploit vulnerabilities and access sensitive information.

- **Describe Experimental Setup:**

  Detail the experimental setup conducted to develop a model for detecting XSS attacks using the Multinomial Naive Bayes classifier.

- **Evaluate Performance of Machine Learning Model:**

  Assess the effectiveness of the Multinomial Naive Bayes classifier in detecting XSS attacks, considering factors such as accuracy and efficiency.

- **Conclude with Implications and Future Directions:**

  Conclude by discussing the implications of the study's findings and suggesting potential avenues for future research and enhancement of XSS detection methodologies.

By achieving these objectives, the project aims to revolutionize the landscape of roadside assistance, setting new standards for convenience, reliability, and affordability. Through its commitment to user satisfaction and community development, the project endeavors to not only meet but exceed the expectations of its users, becoming the preferred choice for roadside assistance needs. Additionally, by fostering innovation and embracing technological advancements, the project seeks to adapt to changing market dynamics and remain a leader in the industry for years to come.

1.3 WHY MACHINE LEARNING?

To better understand the uses of machine learning, consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning.

All these examples echo the vital role machine learning has begun to take in today’s data-rich world. Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

The process flow depicted here represents how machine learning works

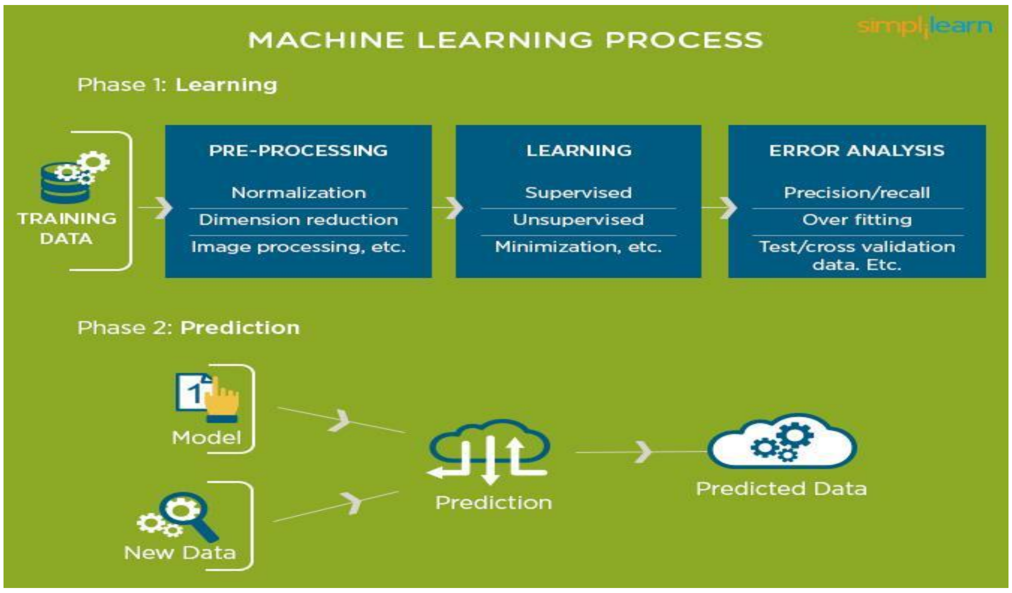


Fig.1.3 Process of Machine Learning

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

1.4 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data. By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning is able to produce accurate results and analysis.

In addition to the mentioned applications, machine learning has found extensive use in various other domains such as personalized recommendation systems in e-commerce platforms, predictive maintenance in industrial settings, fraud detection in financial transactions, and medical diagnosis and prognosis based on patient data analysis. These applications leverage the power of machine learning to uncover hidden patterns and insights within vast datasets, enabling businesses and organizations to make data-driven decisions and improve operational efficiency.

Furthermore, machine learning techniques like deep learning have revolutionized fields like natural language processing, enabling the development of chatbots, virtual assistants, and language translation systems with unprecedented accuracy and fluency. Additionally, reinforcement learning algorithms have been employed in autonomous vehicles to optimize driving behavior and enhance safety on the roads. Overall, machine learning continues to evolve and permeate various aspects of our daily lives, offering solutions to complex problems and driving innovation across industries. Its ability to handle large volumes of data and extract valuable insights has positioned it as a cornerstone of modern data analytics and decision-making processes.

Overall, machine learning continues to evolve and permeate various aspects of our daily lives, offering solutions to complex problems and driving innovation across industries. Its ability to handle large volumes of data and extract valuable insights has positioned it as a cornerstone of modern data analytics and decision-making processes.

2. LITERATURE REVIEW

Title : Machine Learning Based Cross-Site Scripting Detection in Online Social Network.

Author : Rui Wang; Xiaoqi Jia; Qinlei Li; Shengzhi Zhang.

Abstract:

Nowadays online social network (OSN) is one of the most popular internet services in the world. It allows us to communicate with others and share knowledge. However, from the security's point of view, OSN is becoming the favorite target for the attackers, and is under a lot of threats such as cross-site scripting (XSS) attacks. In this paper, we present a novel approach using machine learning to do XSS detection in OSN. Firstly, we leverage a new method to capture identified features from web pages and then establish classification models which can be used in XSS detection. Secondly, we propose a novel method to simulate XSS worm spreading and build our webpage database. Finally, we set up experiments to verify the classification models using our test database. Our experiment results demonstrate that our approach is an effective countermeasure to detect the XSS attack.safety and convenience of motorists on the road.

Title : Classification of XSS Attacks by Machine Learning with Frequency of Appearance and Co-occurrence.

Author : Sota Akaishi; Ryuya Uda.

Abstract:

Nowadays, Sentiment Analysis has become an active research area due to the availability of many opinionated data through increased activity in Blogging, Tagging, Podcasting, social networking sites, RSS feeds, and Social Bookmarking. In the present situation, the whole world is facing the crisis of the COVID-19 pandemic. Particularly, let's talk about nationwide lockdown in India to control the spread of COVID-19. The government relies on social media to observe people's aviews on their policies during the lockdown. In this paper, Twitter data has been used for Sentiment Analysis, which focus on people opinion during the COVID-19 nationwide Lockdown effect in India. Different keywords data was collected on various dates between March 25, 2020, to June 09, 2020. This research work is an application of the real-time TextBlob sentiment analyzer tool built based on the Natural Language Toolkit (NLTK). Relevant keyword tweets were extracted by tweeter API. Then a model was trained to classify the result on a specific opinion. This NLPbased sentiment analysis model is ideal for analyzing the emotions while tested with seven primary keywords: Lockdown1.0, Migrant Workers, Indian Economic, ICMR, Lockdown5.0, Medical Facilities, and

Police. The result shows that Lockdown 1.0 got the most positive sentiments, followed by ICMR and Medical Facility.

Title : Sentiment Analysis of Lockdown in India During COVID-19: A Case Study on Twitter.

Author : Prasoon Gupta; Sanjay Kumar; R. R. Suman; Vinay Kumar.

Abstract:

With the rapid increase in the use of the Internet, sentiment analysis has become one of the most popular fields of natural language processing (NLP). Using sentiment analysis, the implied emotion in the text can be mined effectively for different occasions. People are using social media to receive and communicate different types of information on a massive scale during COVID-19 outburst. Mining such content to evaluate people's sentiments can play a critical role in making decisions to keep the situation under control. The objective of this study is to mine the sentiments of Indian citizens regarding the nationwide lockdown enforced by the Indian government to reduce the rate of spreading of Coronavirus. In this work, the sentiment analysis of tweets posted by Indian citizens has been performed using NLP and machine learning classifiers.

From April 5, 2020 to April 17, 2020, a total of 12 741 tweets having the keywords "Indialockdown" are extracted. Data have been extracted from Twitter using Tweepy API, annotated using TextBlob and VADER lexicons, and preprocessed using the natural language tool kit provided by the Python. Eight different classifiers have been used to classify the data. The experiment achieved the highest accuracy of 84.4% with Linear SVC classifier and unigrams. This study concludes that the majority of Indian citizens are supporting the decision of the lockdown implemented by the Indian government during corona outburst.

3. SYSTEM DESIGN

3.1 EXISTING SYSTEM:

In previous, The project is based on support vector method, the k-nearest neighbor method, and the random forest method. The existing system relies on support vector machines (SVM), k-nearest neighbor (KNN), and random forest methods for addressing machine learning tasks. However, these approaches exhibit limitations.

SVM struggles with large datasets and noisy data, particularly when target classes overlap or when feature dimensionality exceeds the number of training samples. KNN's performance suffers from computational inefficiency, especially with high-dimensional data, due to the costly calculation of distances between data instances.

While random forests offer improvements over single decision trees, they may lack predictive accuracy on complex problems compared to more advanced techniques like gradient-boosted trees. Additionally, the interpretability of random forests is compromised by their ensemble nature. These drawbacks underscore the need for exploring alternative methodologies to enhance performance and scalability in machine learning applications.

3.1.1 DISADVANTAGES:

- SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

- Does not work well with large dataset as calculating distances between each data instance would be very costly and does not work well with high dimensionality as this will complicate the distance calculating process to calculate distance for each dimension. Sensitive to noisy and missing data.

- Random forests can be an improvement on single decision trees, more sophisticated techniques are available. Prediction accuracy on complex problems is usually inferior to gradient-boosted trees. A forest is less interpretable than a single decision tree.

- While random forests offer robustness against overfitting and improved performance over single decision trees, they still may not provide the highest prediction accuracy on complex problems compared to more advanced techniques like gradient-boosted trees. Gradient-boosted trees often outperform random forests in scenarios where intricate patterns and relationships exist within the data.

3.2 PROPOSED SYSTEM:

The proposed system leverages the Multinomial Naive Bayes (MNB) classifier for detecting cross-site scripting (XSS) attacks in web applications. MNB, renowned for its effectiveness in natural language processing tasks, offers several advantages for this application.

Firstly, its implementation is quick and requires minimal computational resources, making it ideal for real-time detection scenarios. Secondly, MNB is computationally efficient, enabling it to handle large datasets efficiently. This capability is crucial given the abundance of features typically encountered in web application data.

Additionally, MNB's ability to handle a large number of features makes it particularly well-suited for text classification tasks, aligning seamlessly with the nature of XSS attack detection, where inputs often comprise numerous words and phrases. Overall, the proposed system harnesses the strengths of MNB to provide a robust and efficient solution for detecting XSS attacks, contributing to the enhancement of web application security.

3.2.1 ADVANTAGES:

- It can be implemented quickly with minimal computational resources.Fast and efficient: MNB is computationally efficient and can handle large datasets with many features.

- Handles large number of features: MNB can handle a large number of features, which makes it particularly useful for text classification tasks, where the input data consists of a large number of words.

- Robust to irrelevant features: Multinomial Naive Bayes (MNB) is robust to irrelevant features in the dataset, meaning it can still perform well even if some of the input features are not informative or relevant to the classification task at hand. This property makes MNB suitable for tasks where the input data may contain noise or extraneous information.

- Works well with sparse data: MNB performs well even when the input data is sparse, meaning it has a high number of zero or near-zero values. This makes it particularly useful for text classification tasks where the feature space is typically sparse, as documents often contain only a small subset of all possible words.

4. BACKGROUND STUDY

4.1 CLASSIFICATION OF MACHINE LEARNING:

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model of a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. Semi-Supervised Learning algorithms develop mathematical models from incomplete training data, where a portion of the sample inputs are missing the desired output.

Classification algorithms and Regression Algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values true and false. Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

Another category of machine learning tasks is unsupervised learning, where the algorithm is given a dataset without explicit instructions on what to do with it. Instead, it must find patterns and structures within the data on its own. Clustering algorithms, such as K-means clustering, are commonly used in unsupervised learning to group similar data points together based on their features. This can be useful for tasks like customer segmentation in marketing or identifying patterns in biological data. Another type of unsupervised learning is dimensionality reduction, which aims to reduce the number of features in a dataset while preserving its important information. Techniques like Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are often used for this purpose, facilitating visualization and analysis of high-dimensional data.

4.2 SUPERVISED LEARNING:

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X) . The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Trees and support vector machines. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics.

Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification.

A regression problem is when the output variable is a real or continuous value, such as "salary" or "weight". Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points. This type of learning is commonly used in autonomous systems, game playing, and robotics, where the agent learns to optimize its actions over time to achieve a specific goal.

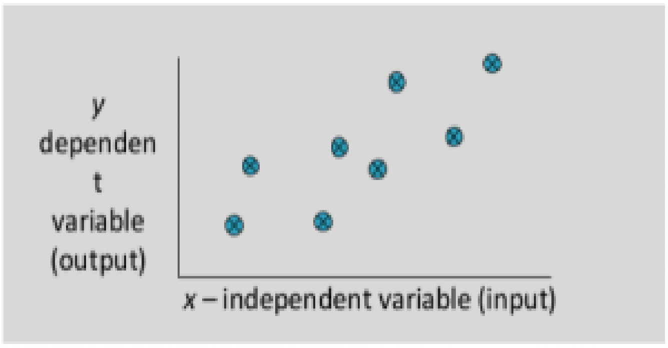


Fig.4.2 Supervised Learning

4.3 UNSUPERVISED LEARNING:

In unsupervised learning, the algorithm builds a mathematical model of a set of data which contains only inputs and no desired outputs. Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in feature learning. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.

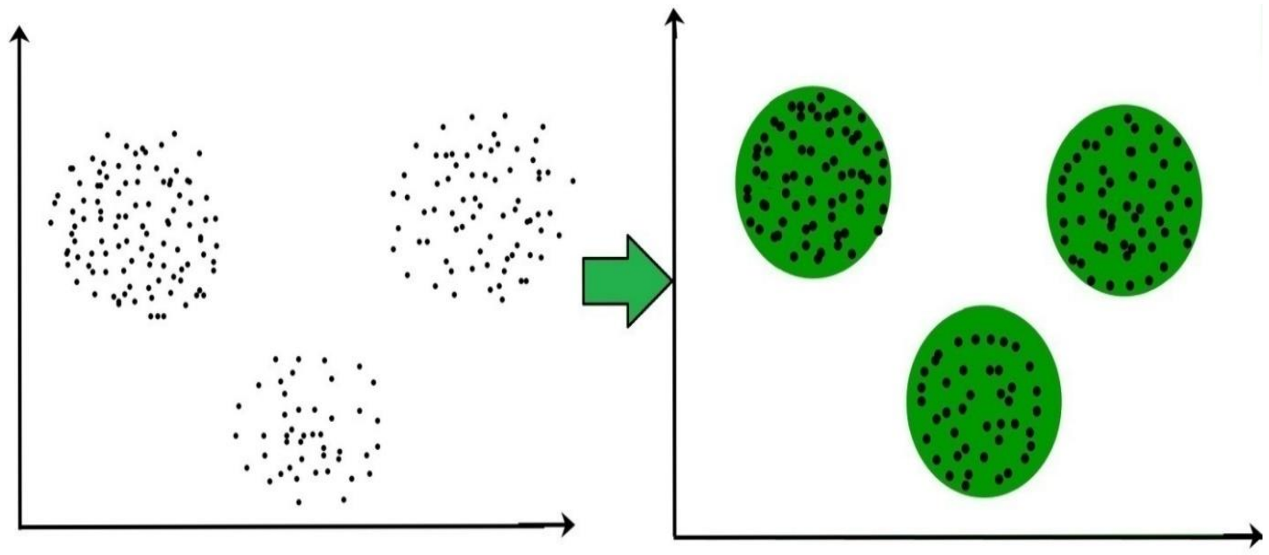


Fig.4.3.1 Unsupervised Learning

It is not necessary for clusters to be a spherical. Such as :

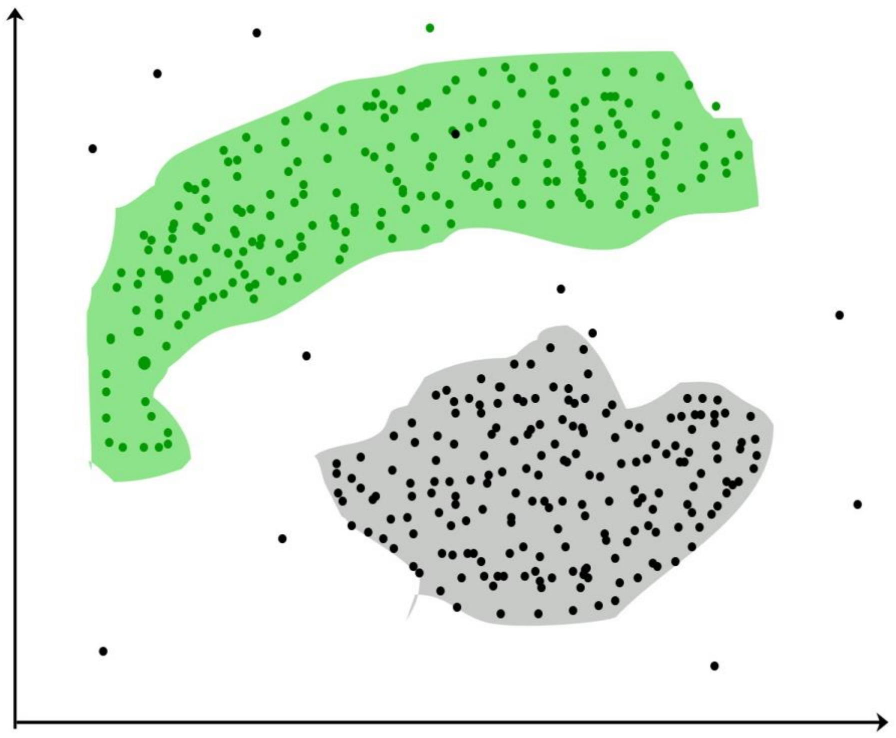


Fig.4.3.2 Clustering Image

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for calculation of the outliers.

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters. Clustering algorithms vary in their assumptions about the underlying structure of the data and the definition of similarity between data points. For example, K-means clustering assumes that clusters are spherical and of equal size, while hierarchical clustering does not make any explicit assumptions about cluster shape or size. Density-based clustering algorithms, such as DBSCAN, identify clusters based on regions of high data density, allowing for more flexible cluster shapes.s

4.4 CLUSTERING ALGORITHMS:

K-means clustering algorithm – It is the simplest unsupervised learning algorithm that solves clustering problem.K-means algorithm partition n observations into k clusters where each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster. Call Up a Friend or Relative for Help:

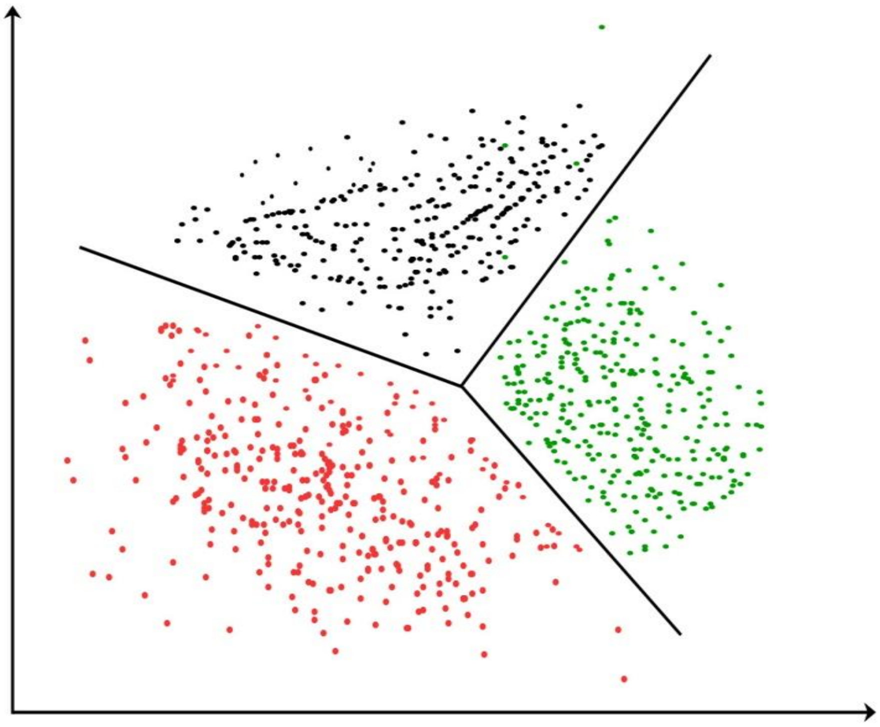


Fig.4.4 K-Means Clustering

Applications of Clustering in different fields

1. Marketing : It can be used to characterize & discover customer segments for marketing purposes.

2. Biology : It can be used for classification among different species of plants and animals.

3. Libraries : It is used in clustering different books on the basis of topics and information.

4. Insurance : It is used to acknowledge the customers, their policies and identifying the frauds.

5. City Planning : It is used to make groups of houses and to study their values based on their geographical locations and other factors present.

6. Earthquake studies : By learning the earthquake affected areas we can determine the dangerous zones.

4.5 APPLICATION OF MACHINE LEARNING:

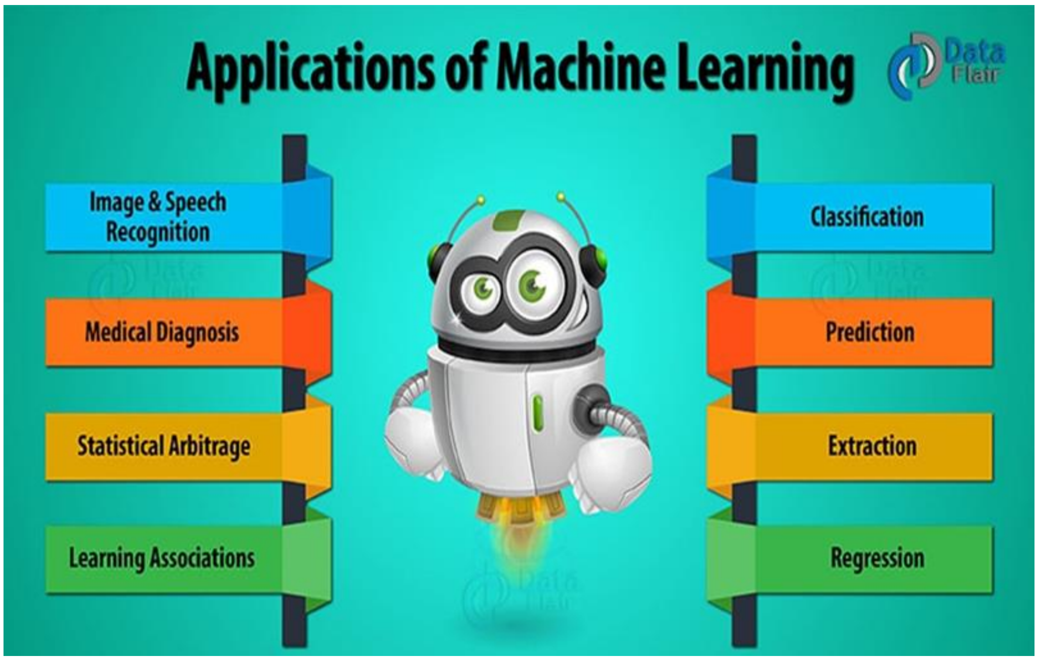


Fig.4.5.1 APPLICATIONS OF MACHINE LEARNING

4.5.1 MACHINE LEARNING IN HEALTHCARE:

Doctors and medical practitioners will soon be able to predict with accuracy on how long patients with fatal diseases will live. Medical systems will learn from data and help patients save money by skipping unnecessary tests. Radiologists will be replaced by machine learning algorithmsOffer guidance on how to search for a qualified mechanic or auto technician using online resources, directories, or mobile apps. Include criteria for selecting a reputable mechanic, such as certifications, customer reviews, and proximity to the user's location.

BERG
Back to Biology for a Healthier Tomorrow
Pfizer
Machine Learning Applications in Healthcare
health MAP
Genentech
A Member of the Roche Group

Fig.4.5.2 HEALTHCARE

McKinsey Global Institute estimates that applying machine learning techniques to better inform decision making could generate up to $100 billion in value based on optimized innovation, enhanced efficiency of clinical trials and the creation of various novel tools for physicians, insurers and consumers. Computers and Robots cannot replace doctors or nurses, however the use of life-saving technology (machine learning) can definitely transform healthcare domain. When we talk about efficiency of machine learning, more data produces effective results – and the healthcare industry is residing on a data goldmine.

i) Drug Discovery/Manufacturing

Manufacturing or discovering a new drug is expensive and lengthy process as thousands of compounds need to be subjected to a series of tests, and only a single one might result in a usable drug. Machine learning can speed up one or more of these steps in this lengthy multi-step process.

Machine Learning Examples in Healthcare for Drug Discovery. Pfizer is using IBM Watson on its immuno-oncology (a technique that uses body's immune system to help fight cancer) research. This is one of the most significant uses of IBM Watson for drug discovery. Pfizer has been using machine learning for years to sieve through the data to facilitate research in the areas of drug discovery (particularly the combination of multiple drugs) and determine the best participant for a clinical trial.

ii) Personalized Treatment/Medication

Imagine when you walk in to visit your doctor with some kind of an ache in your stomach. After snooping into your symptoms, the doctor inputs them into the computer that extracts the latest research that the doctor might need to know about how to treat your ache. You have an MRI and a computer helps the radiologist detect problems that possibly could be too small for the human eye to see. In the end, a computer scans all your health records and family medical history and compares it to the latest research to advice a treatment protocol that is particularly tailored to your problem. Machine learning is all set to make a mark in personalized care.

Personalized treatment has great potential for growth in future, and machine learning could play a vital role in finding what kind of genetic makers and genes respond to a particular treatment or medication. Personalized medication or treatment based on individual health records paired with analytics is a hot research area as it provides better disease assessment. In future, increased usage of sensor integrated devices and mobile apps with sophisticated remote monitoring and health-measurement capabilities, there would be another data deluge that could be used for treatment.

4.6 PRINCIPAL COMPONENT ANALYSIS:

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

Importantly, the dataset on which PCA technique is to be used must be scaled. The results are also sensitive to the relative scaling. As a layman, it is a method of summarizing data. Imagine some wine bottles on a dining table. Each wine is described by its attributes like colour, strength, age, etc. But redundancy will arise because many of them will measure related properties. So what PCA will do in this case is summarize each wine in the stock with less characteristics.

Intuitively, Principal Component Analysis can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint. Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science. Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, while being used in different business, science, and social science domains. In today's business, data analysis is playing a role in making decisions more scientific and helping the business achieve effective operation.

Principal Component Analysis (PCA) serves as a powerful tool in data science by reducing the dimensionality of complex datasets while retaining as much variance as possible. By transforming high-dimensional data into a lower-dimensional space, PCA enables visualization and interpretation of data patterns in a more manageable form. This "projection" or "shadow" provided by PCA allows users to identify the most informative features and gain insights into the underlying structure of the data.

5. METHODOLOGY

5.1 DATA PRE-PROCESSING:

To get a model with good accuracy, you need to assemble and prepare the source dataset correctly. The data set must contain malicious and benign JavaScript code. The data set for the experiment consists of 240,000 instances of malicious and benign code. Malicious code is 40,000 instance [7], and benign code is 200,000 instance.

5.2 DATA COLLECTION:

The source data set is represented as queries with a different number of parameters. Each request is decoded into Unicode characters. After decoding, parameters of this query are extracted using regular expressions. Abnormal queries containing a large number of parameters are removed from the dataset.

5.3 TRAINING DATA AND TEST DATA:

For choosing a model we split our dataset into train and test. Here data's are split into 3:1 ratio that means Training data having 70 percent and testing data having 30 percent. In this split process preforming based on train_test_split model. After splitting we get xtrain xtest and ytrain ytest.

5.4 MODEL CREATION:

Contextualise machine learning in your organisation. Explore the data and choose the type of algorithm. Prepare and clean the dataset. Split the prepared dataset and perform cross validation. Perform machine learning optimization. Deploy the model.

5.5 MODEL PREDICTION:

Predictive modeling is a statistical technique using machine learning and data mining to predict and forecast likely future outcomes with the aid of historical and existing data. It works by analyzing current and historical data and projecting what it learns on a model generated to forecast likely outcomes. In this Project, our final prediction is to predict whether a script should be safe or malicious.

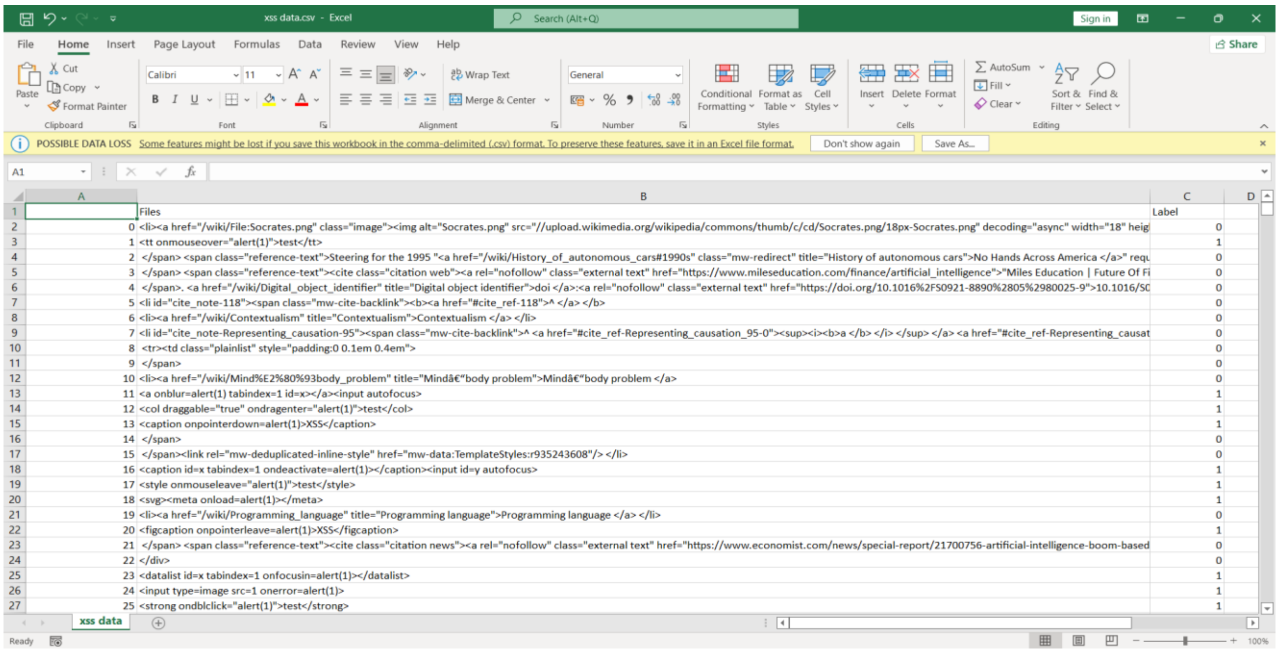
6. MODULES

6.1 DATA PREPARATION AND FEATURE ENGINEERING:

In this module, the dataset 'xss data.csv' is read using Pandas, and the data is prepared for model training. The 'Files' column contains text data representing web pages, while the 'Label' column contains the target variable indicating whether each page is benign or XSS. To enable machine learning algorithms to process the text data, the CountVectorizer from scikit-learn is utilized. This converts the text data into numerical feature vectors, facilitating analysis by the algorithms.

Subsequently, the data is split into training and testing sets using train_test_split from scikit-learn. This step is crucial for evaluating the model's performance, as it allows for validation on unseen data. By ensuring the separation of training and testing data, the model's ability to generalize to new instances can be accurately assessed. After splitting the data, the numerical feature vectors obtained from CountVectorizer are fed into the machine learning model, along with their corresponding labels. The model is then trained on the training set using fit() function.

Once trained, the model's performance is evaluated on the testing set using various metrics such as accuracy, precision, recall, and F1-score.Furthermore, to enhance the model's performance and prevent overfitting, hyperparameter tuning techniques such as grid search or randomized search can be employed.



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B	C
1		Files	Label
2	0	<li><a href="/wiki/File:Socrates.png" class="image"><img alt="Socrates.png" src="//upload.wikimedia.org/wikipedia/commons/thumb/c/cd/Socrates.png/18px-Socrates.png" decoding="async" width="18" heig	0
3	1	<tt onmouseover="alert(1)">test</tt>	1
4	2	</span> <span class="reference-text">Steering for the 1995 "<a href="/wiki/History_of_autonomous_cars#1990s" class="mw-redirect" title="History of autonomous cars">No Hands Across America </a>" requ	0
5	3	</span> <span class="reference-text"><cite class="citation web"><a rel="nofollow" class="external text" href="https://www.mileseducation.com/finance/artificial_intelligence">"Miles Education | Future Of Fi	0
6	4	</span>. <a href="/wiki/Digital_object_identifier" title="Digital object identifier">doi </a>:<a rel="nofollow" class="external text" href="https://doi.org/10.1016%2FS0921-8890%2805%2980025-9">10.1016/SC	0
7	5	<li id="cite_note-118"><span class="mw-cite-backlink"><b><a href="#cite_ref-118">^ </a> </b>	0
8	6	<li><a href="/wiki/Contextualism" title="Contextualism">Contextualism </a> </li>	0
9	7	<li id="cite_note-Representing_causation-95"><span class="mw-cite-backlink">^ <a href="#cite_ref-Representing_causation_95-0">^{<i><b>a </b> </i>} </a> <a href="#cite_ref-Representing_causat	0
10	8	<tr><td class="plainlist" style="padding:0 0.1em 0.4em">	0
11	9	</span>	0
12	10	<li><a href="/wiki/Mind%E2%80%93body_problem" title="Mindâ€“body problem">Mindâ€“body problem </a>	0
13	11	<a onblur=alert(1) tabindex=1 id=x></a><input autofocus>	1
14	12	<col draggable="true" ondragenter="alert(1)">test</col>	1
15	13	<caption onpointerdown=alert(1)>XSS</caption>	1
16	14	</span>	0
17	15	</span><link rel="mw-deduplicated-inline-style" href="mw-data:TemplateStyles:r935243608"/> </li>	0
18	16	<caption id=x tabindex=1 ondeactivate=alert(1)></caption><input id=y autofocus>	1
19	17	<style onmouseleave="alert(1)">test</style>	1
20	18	<svg><meta onload=alert(1)></meta>	1
21	19	<li><a href="/wiki/Programming_language" title="Programming language">Programming language </a> </li>	0
22	20	<figcaption onpointerleave=alert(1)>XSS</figcaption>	1
23	21	</span> <span class="reference-text"><cite class="citation news"><a rel="nofollow" class="external text" href="https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based	0
24	22	</div>	0
25	23	<datalist id=x tabindex=1 onfocusin=alert(1)></datalist>	1
26	24	<input type=image src=1 onerror=alert(1)>	1
27	25	<strong ondblclick="alert(1)">test</strong>	1

Fig.6.1 Data Preparation and Feature Engineering

6.2 MODEL TRAINING AND EVALUATION:

In this module, a Multinomial Naive Bayes classifier (MultinomialNB) is trained on the training data (X_train, y_train). This involves utilizing the features extracted from the training dataset (X_train) along with corresponding labels (y_train) to train the classifier. Following training, the performance of the trained classifier is evaluated on the test data (X_test, y_test) to assess its accuracy and effectiveness in detecting XSS attacks. This evaluation is crucial for understanding how well the classifier generalizes to unseen data.

To comprehensively assess the classifier's performance, various evaluation metrics such as accuracy_score, classification_report, and confusion_matrix are computed. These metrics provide insights into the classifier's ability to correctly classify benign and malicious instances, identify potential areas of improvement, and gauge overall effectiveness in detecting XSS attacks.By leveraging these evaluation metrics, stakeholders can make informed decisions regarding the deployment and optimization of the XSS detection system, ultimately enhancing the security posture of web applications against potential threats.

In addition to evaluating the classifier's performance using standard metrics, it's also important to consider the implications of false positives and false negatives in the context of XSS detection. False positives occur when the classifier incorrectly identifies benign web pages as malicious, potentially leading to unnecessary security alerts or disruptions for legitimate users.

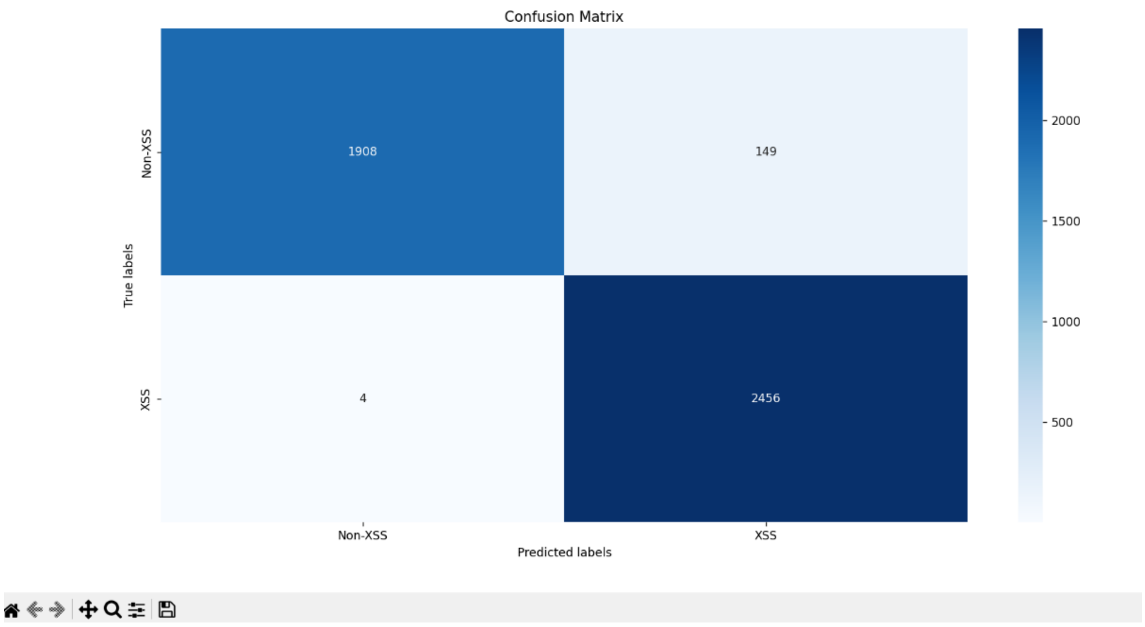
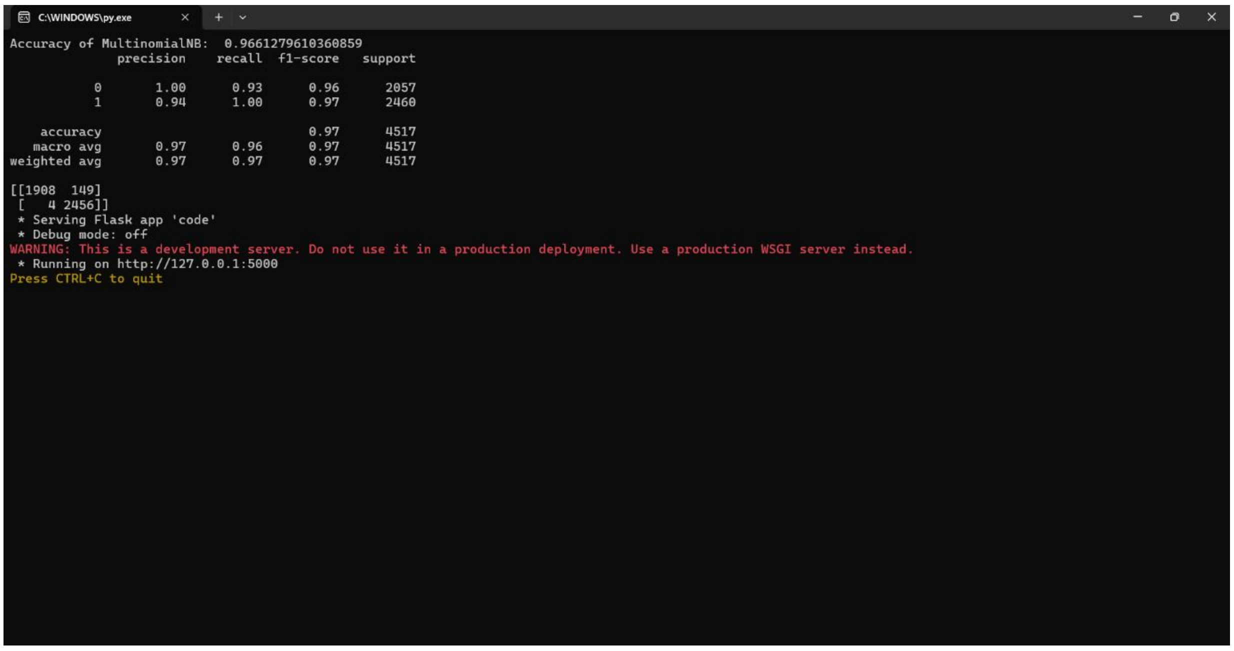


Fig.6.2 Confusion matrix

6.3 MODEL PERSISTANCE:

The MultinomialNB classifier is saved to a file named 'NB_XSS_model.pkl' using the joblib.dump function. This action ensures that the trained model can be stored and reused later without the necessity for retraining. By saving the model in this manner, efficiency and scalability are enhanced, as it eliminates the need to repeatedly train the model from scratch for future use cases. This streamlined approach facilitates seamless integration of the trained model into production environments, where it can be readily deployed to make predictions on new data, thereby optimizing operational efficiency and resource utilization.

Saving the MultinomialNB classifier to a file named 'NB_XSS_model.pkl' using joblib.dump ensures the preservation of the trained model's state, including its parameters and learned patterns, for future use without the need for retraining. This practice not only promotes efficiency and scalability but also facilitates seamless integration into production environments. By eliminating the necessity for repeated training, resources are conserved, and operational efficiency is optimized. Furthermore, the stored model can be easily deployed to make predictions on new data, enhancing the responsiveness and agility of the XSS detection system in real-world scenarios. Overall, leveraging joblib.dump for model persistence enhances the usability, maintainability, and performance of the XSS detection solution, contributing to a more robust and effective defense against potential threats.



```
Accuracy of MultinomialNB:  0.9661279610360859
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      2057
           1       0.94      1.00      0.97      2460

    accuracy                           0.97      4517
   macro avg       0.97      0.96      0.97      4517
weighted avg       0.97      0.97      0.97      4517

[[1908  149]
 [   4 2456]]
 * Serving Flask app 'code'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Fig.6.3 Model Persistance

6.4 Web Application Development:

Flask, a lightweight web framework for Python, serves as the deployment platform for the trained model, transforming it into a web application. The Flask application comprises two routes: '/' serving as the home page and '/predict' for handling input data and generating predictions. When a user submits text input via a form, the Flask application receives the input, preprocesses it, applies the CountVectorizer transformation to convert it into a numerical feature vector, and then employs the trained model to predict whether the input contains an XSS attack.

Following prediction, the result is presented to the user via a designated result template ('result.html'). This template displays the prediction outcome, effectively informing the user whether the submitted text is indicative of an XSS attack or not. This seamless integration of Flask with the trained model provides users with a user-friendly interface for interacting with the XSS detection system, enhancing overall usability and accessibility.

To interact with the XSS detection system, enabling them to quickly and conveniently assess the security status of their input text. Additionally, Flask's simplicity and flexibility make it an ideal choice for deploying machine learning models as web applications, allowing for easy customization and extension of functionality as needed. By leveraging Flask's routing and templating capabilities, developers can create intuitive and visually appealing interfaces that enhance user experience while maintaining the robustness and reliability of the underlying machine learning model.

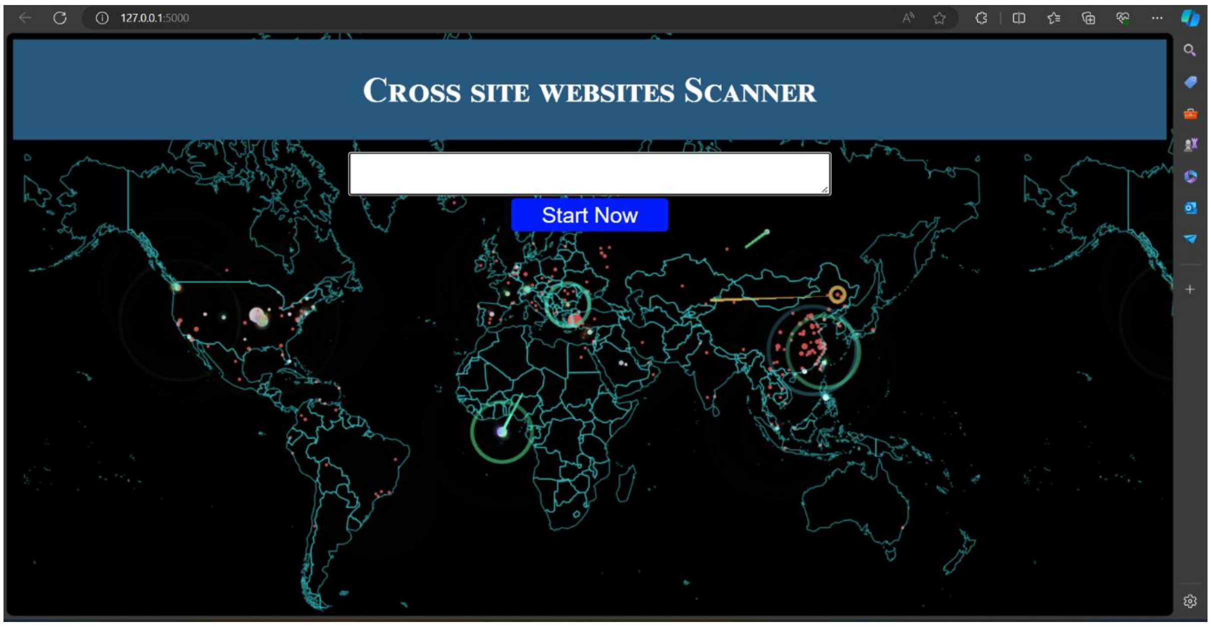


Fig.6.4 Web Application Deployment Result

7. ALGORITHM IMPLEMENTATION

7.1 MULTINOMIAL NAÏVE BAYES ALGORITHM:

The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance.

With an ever-growing amount of textual information stored in electronic form such as legal documents, policies, company strategies, etc., automatic text classification is becoming increasingly important. This requires a supervised learning technique that classifies every new document by assigning one or more class labels from a fixed or predefined class. It uses the bag of words approach, where the individual words in the document constitute its features, and the order of the words is ignored. This technique is different from the way we communicate with each other. It treats the language like it's just a bag full of words and each message is a random handful of them. Large documents have a lot of words that are generally characterized by very high dimensionality feature space with thousands of features. Hence, the learning algorithm requires to tackle high dimensional problems, both in terms of classification performance and computational speed.

Naïve Bayes, which is computationally very efficient and easy to implement, is a learning algorithm frequently used in text classification problems. Two event models are commonly used:

- Multivariate Bernoulli Event Model
- Multivariate Event Model

The Multivariate Event model is referred to as Multinomial Naive Bayes. When most people want to learn about Naive Bayes, they want to learn about the Multinomial Naive Bayes Classifier. However, there is another commonly used version of Naïve Bayes, called Gaussian Naive Bayes Classification. Check out the free course on naive Bayes supervised learning.

Naive Bayes is based on Bayes' theorem, where the adjective Naïve says that features in the dataset are mutually independent. Occurrence of one feature does not affect the probability of occurrence of the other feature. For small sample sizes, Naïve Bayes can outperform the most powerful alternatives. Being relatively robust, easy to implement, fast, and accurate, it is used in many different fields. Check out naïve bayes classifiers.

For Example, Spam filtering in email, Diagnosis of diseases, making decisions about treatment, Classification of RNA sequences in taxonomic studies, to name a few. However, we have to keep in mind about the type of data and the type of problem to be solved that dictates which classification model we want to choose. Strong violations of the independence assumptions and non-classification problems can lead to poor performance. In practice, it is recommended to use different classification models on the same dataset and then consider the performance, as well as computational efficiency.

To understand how Naïve Bayes works, first, we have to understand the concept of Bayes' rule. This probability model was formulated by Thomas Bayes (1701-1761) and can be written as:

$$Posterior\ Probability = \frac{Conditional\ Probability * Prior\ Probability}{Predictor\ Prior\ Probability}$$

$$P\left(\frac{A}{B}\right) = \left(\frac{P(A \cap B)}{P(B)}\right) = \frac{P(A) * P\left(\frac{B}{A}\right)}{P(B)}$$

where,

PA= the prior probability of occurring A

PBA= the condition probability of B given that A occurs

PAB= the condition probability of A given that B occurs

PB= the probability of occuring B

The Multivariate Event model is referred to as Multinomial Naive Bayes.When most people want to learn about Naive Bayes, they want to learn about the Multinomial Naive Bayes Classifier. However, there is another commonly used version of Naïve Bayes, called Gaussian Naive Bayes Classification. Check out the free course on naive Bayes supervised learning.

7.2 COUNTVECTORIZER:

CountVectorizer is a machine learning tool used for natural language processing and text classification tasks. It is a commonly used tool in the context of web application security, specifically in cross-site scripting (XSS) detection and prevention.XSS is a type of attack where an attacker injects malicious scripts into a web page viewed by unsuspecting users. These scripts can be used to steal sensitive information or perform other malicious actions.

To prevent XSS attacks, web applications can use input validation and output encoding to filter out malicious scripts. CountVectorizer can be used in this context to automatically identify and categorize potentially malicious scripts by analyzing the text content of the web page. Collect a set of web pages, some of which are known to be vulnerable to XSS attacks and some of which are not. Preprocess and clean the text data from the web pages by removing stop words, non-alphabetic characters, and converting all text to lowercase. Initialize the CountVectorizer and fit it to the preprocessed text data. Transform the text data into a matrix of word counts using the CountVectorizer's transform method. Train a machine learning classifier (such as a logistic regression model) on the word frequency matrix and the labels (vulnerable or not vulnerable).

Use the trained classifier to predict the vulnerability of new web pages by transforming their text data into a word frequency matrix and applying the classifier's predict method. By using CountVectorizer in conjunction with a machine learning classifier, web applications can automatically detect and prevent XSS attacks without relying solely on manual input validation and output encoding. However, it's important to note that machine learning models can never be 100% accurate, and attackers can always find new and creative ways to bypass detection techniques. Therefore, it's important to use CountVectorizer as one part of a multi-layered approach to web application security.

While CountVectorizer and machine learning classifiers offer an automated approach to XSS detection and prevention, it's crucial to acknowledge that no solution is foolproof. Attackers may continuously develop new evasion techniques, highlighting the need for a multi-layered security approach that combines automated detection methods with manual input validation and output encoding. By integrating CountVectorizer as part of a comprehensive security strategy, web applications can enhance their resilience against XSS attacks and safeguard sensitive user information more effectively.

7.3 FLOW CHART:

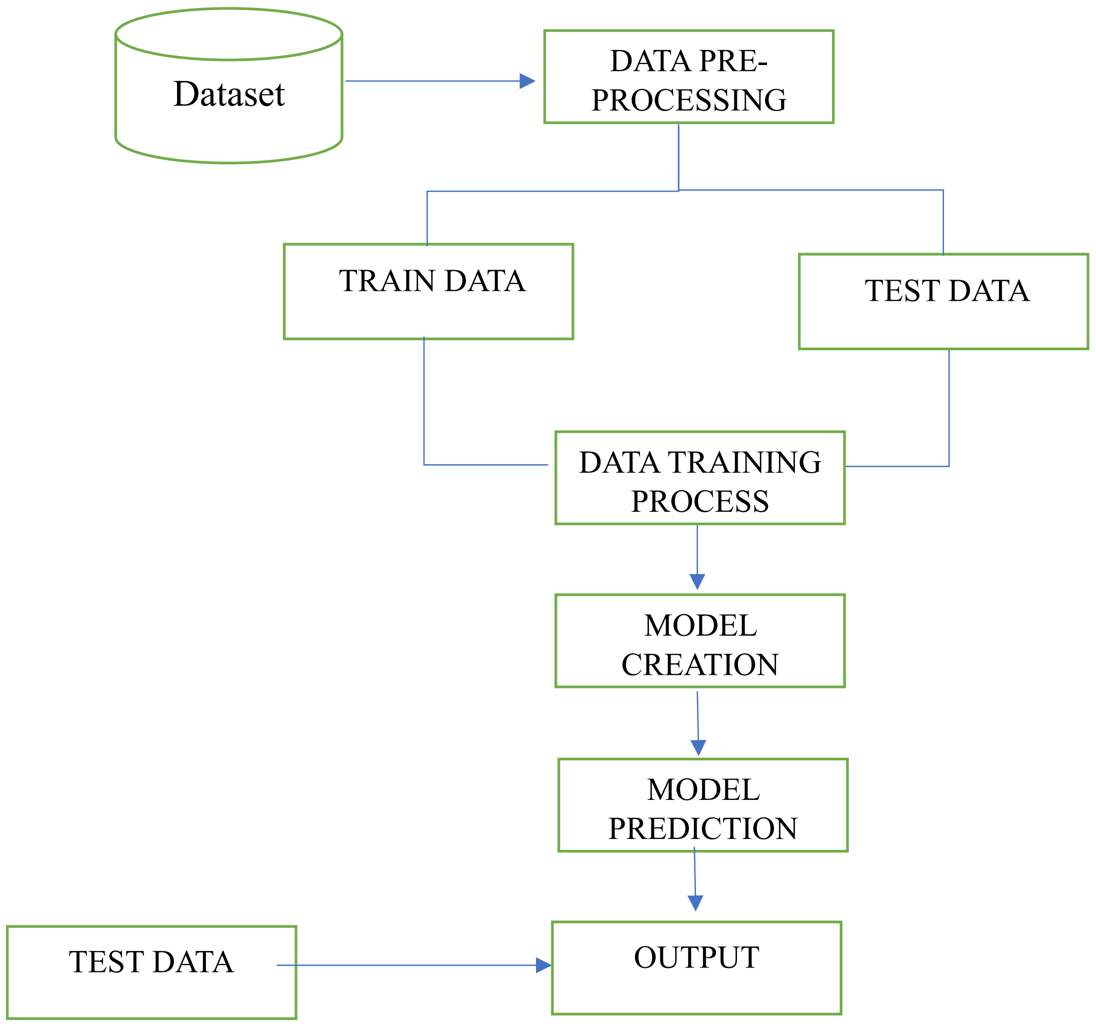


Fig.7.3 Flow Chart

Dataset: This is the initial collection of data that serves as the foundation for the machine learning process. The dataset typically consists of features (independent variables) and labels (dependent variables) that are used to train and evaluate the machine learning model.

Dataset Preprocessing: In this step, the dataset undergoes various preprocessing techniques to clean and prepare the data for training. This may include tasks such as handling missing values, removing duplicates, scaling features, and encoding categorical variables.

Train Data / Test Data Split: After preprocessing, the dataset is divided into two subsets: the training data and the test data. The training data is used to train the machine learning model, while the test data is kept separate and used to evaluate the model's performance.

Data Training Process: In this step, the training data is fed into the machine learning algorithm to train the model. During training, the algorithm learns patterns and relationships in the data that enable it to make predictions or classifications.

Model Creation: Once trained, the machine learning algorithm creates a model based on the patterns and relationships it learned from the training data. This model encapsulates the knowledge gained during the training process and can be used to make predictions on new, unseen data.

Model Prediction: With the model in place, it can now be used to make predictions or classifications on new data. This typically involves inputting the features of the new data into the model and obtaining the predicted output based on the learned patterns.

Output: The output of the model prediction step is the predicted values or classifications for the new data. This output provides insights or predictions based on the model's learned behavior and can be used for decision-making or further analysis.

Test Data: Finally, the performance of the model is evaluated using the test data, which was held out from the training process. By comparing the model's predictions on the test data to the actual labels, metrics such as accuracy, precision, recall, and F1-score can be calculated to assess the model's performance and generalization ability.

7.4 SYSTEM REQUIREMENTS:

7.4.1 HARDWARE REQUIREMENTS:

- RAM : 4GB or 8GB
- Processor : i3 or i5

7.4.2 SOFTWARE REQUIREMENTS:

- Python 3.9
- Windows 10

7.5 SOFTWARE DESCRIPTION:

7.5.1 PYTHON:

Python is an open source programming language. Python was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp. Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks. Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

- Web development
- Game programming
- Desktop GUIs
- Scientific programming
- Network programming.

7.5.2 VERSION 3:

Python 3.0 (also called "Python 3000" or "Py3K") was released on December 3, 2008 It was designed to rectify fundamental design flaws in the language—the changes required could not be implemented while retaining full backwards compatibility with the 2.x series, which necessitated a new major version number. The guiding principle of Python 3 was: "reduce feature duplication by removing old ways of doing things".

Python 3.0 was developed with the same philosophy as in prior versions. However, as Python had accumulated new and redundant ways to program the same task, Python 3.0 had an emphasis on removing duplicative constructs and modules, in keeping with "There should be one— and preferably only one —obvious way to do it".

Nonetheless, Python 3.0 remained a multi-paradigm language. Coders still had options among object-orientation, structured programming, functional programming and other paradigms, but within such broad choices, the details were intended to be more obvious in Python 3.0 than they were in Python 2.x.

Improvements in this release include:

- An ordered dictionary type
- New unit test features including test skipping, new assert methods, and test discovery
- A much faster io module
- Automatic numbering of fields in the str.format() method
- Float repr improvements backported from 3.x
- Tile support for Tkinter
- A backport of the memoryview object from 3.x
- Set literals.

7.5.3 PACKAGES:

1.NUMPY

2.PANDAS

3.SKLEARN

4.FLASK

7.5.4 NUMPY:

NumPy is a Python package. It stands for &#39;Numerical Python&#39;. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear

7.5.5 PANDAS:

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Pandas deals with the following three data structures −

- Series
- DataFrame

7.5.6 SKLEARN:

Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k- means, random forests and DBSCAN. It is designed to work with Python Numpy and SciPy . The scikit-learn project kicked off as a Google Summer of Code (also known as GSoC) project by David Cournapeau as scikits.learn. It gets its name from "Scikit", a separate third-party extension to SciPy.

Scikit is written in Python (most of it) and some of its core algorithms are written in Cython for even better performance. Scikit-learn is used to build models and it is not recommended to use it for reading, manipulating and summarizing data as there are better frameworks available for the purpose. It is open source and released under BSD license.

7.5.7 FLASK:

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask’s framework is more explicit than Django’s framework and is also easier to learn because it has less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine.

Nowadays, the web frameworks provide routing technique so that user can remember the URLs. It is useful to access the web page directly without navigating from the Home page. It is done through the following route() decorator, to bind the URL to a function.

At its core, Flask embodies the concept of a web application framework, encapsulating a comprehensive collection of modules and libraries that abstract away low-level complexities such as protocol handling and thread management. This abstraction empowers developers to focus on application logic and user experience, rather than getting bogged down by technical intricacies.

Built upon the Web Server Gateway Interface (WSGI) toolkit and utilizing the Jinja2 template engine, Flask embodies modern web development principles while maintaining a lightweight and efficient architecture. WSGI facilitates seamless communication between web servers and Python web applications, ensuring compatibility and interoperability across diverse server environments.

One of Flask's standout features is its routing mechanism, which allows developers to define URL patterns and bind them to corresponding functions using the route() decorator. This routing technique enables intuitive navigation within the web application, empowering users to access specific pages directly without the need for excessive navigation from the homepage. By providing a clean and organized URL structure, Flask enhances user experience and simplifies application usage.

In today's dynamic web development landscape, Flask continues to be a preferred choice for developers seeking agility, simplicity, and scalability. Its minimalist design philosophy, coupled with powerful features like routing, templating, and extension support, makes it well-suited for a wide range of web application projects, from simple prototypes to complex enterprise-grade solutions. As the demand for efficient and user-friendly web applications grows, Flask remains at the forefront, empowering developers to craft compelling digital experiences with ease.

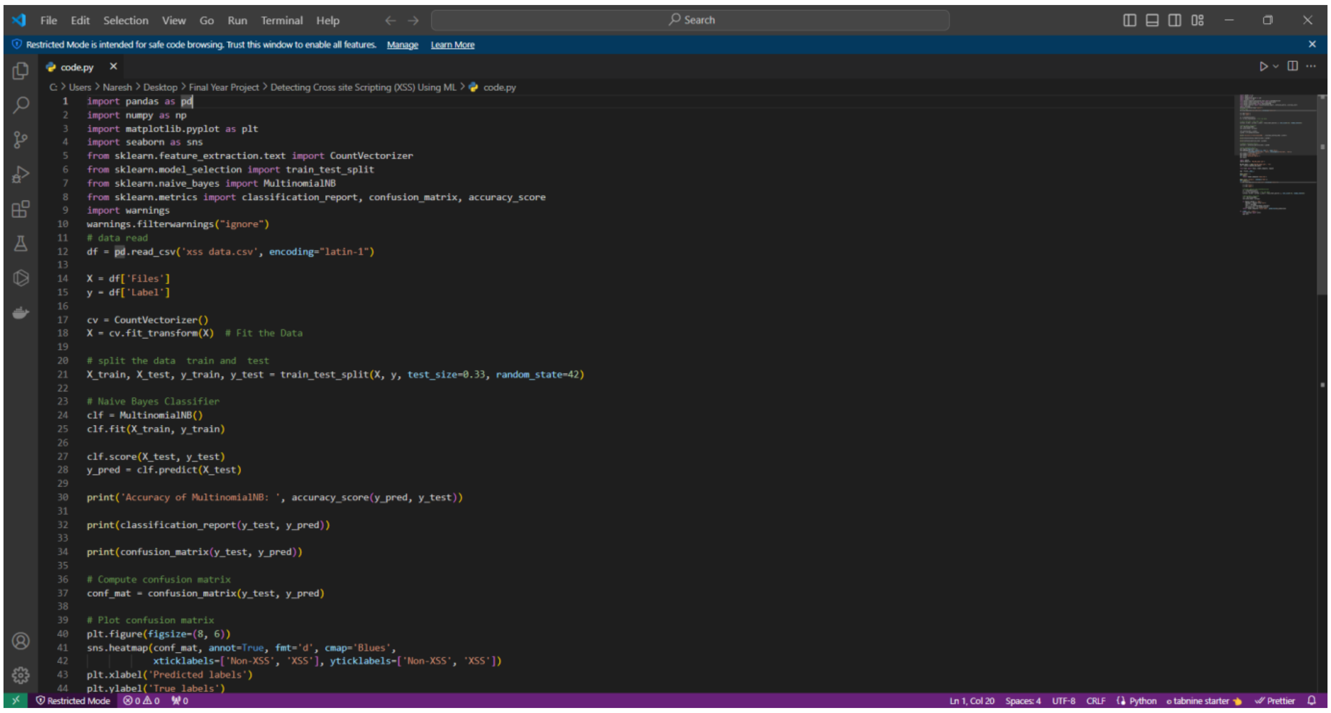
8. APPENDIX

8.1 SOURCE CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
warnings.filterwarnings("ignore")
# data read
df = pd.read_csv('xss data.csv', encoding="latin-1")
X = df['Files']
y = df['Label']
cv = CountVectorizer()
X = cv.fit_transform(X)  # Fit the Data
# split the data  train and  test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
# Naive Bayes Classifier
clf = MultinomialNB()
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
y_pred = clf.predict(X_test)
print('Accuracy of MultinomialNB: ', accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Compute confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
        xticklabels=['Non-XSS', 'XSS'], yticklabels=['Non-XSS', 'XSS'])
```

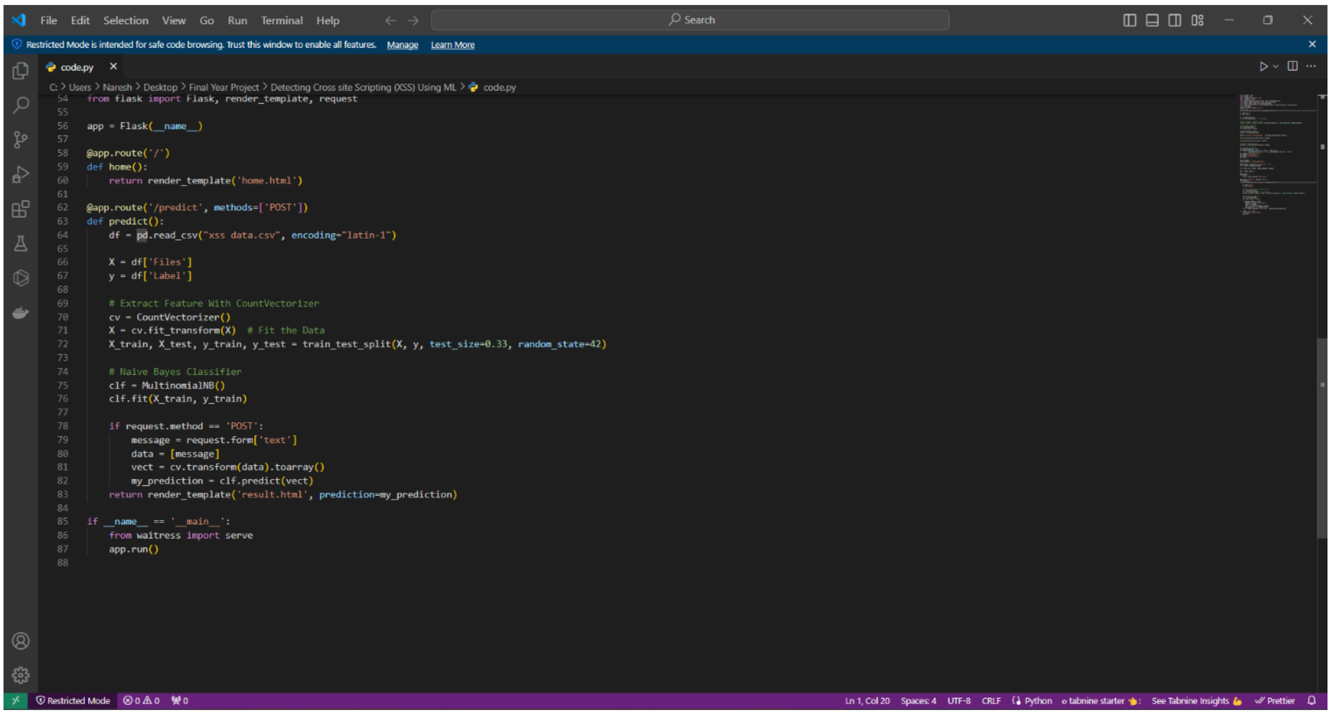
```
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
import joblib
joblib.dump(clf, 'NB_XSS_model.pkl')
NB_XSS_model = open('NB_XSS_model.pkl', 'rb')
clf = joblib.load(NB_XSS_model)
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('home.html')
@app.route('/predict', methods=['POST'])
def predict():
    df = pd.read_csv("xss data.csv", encoding="latin-1")
    X = df['Files']
    y = df['Label']
    # Extract Feature With CountVectorizer
    cv = CountVectorizer()
    X = cv.fit_transform(X)  # Fit the Data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
    # Naive Bayes Classifier
    clf = MultinomialNB()
    clf.fit(X_train, y_train)
    if request.method == 'POST':
        message = request.form['text']
        data = [message]
        vect = cv.transform(data).toarray()
        my_prediction = clf.predict(vect)
    return render_template('result.html', prediction=my_prediction)
if __name__ == '__main__':
    from waitress import serve
    app.run()
```

8.2 SCREENSHOTS:



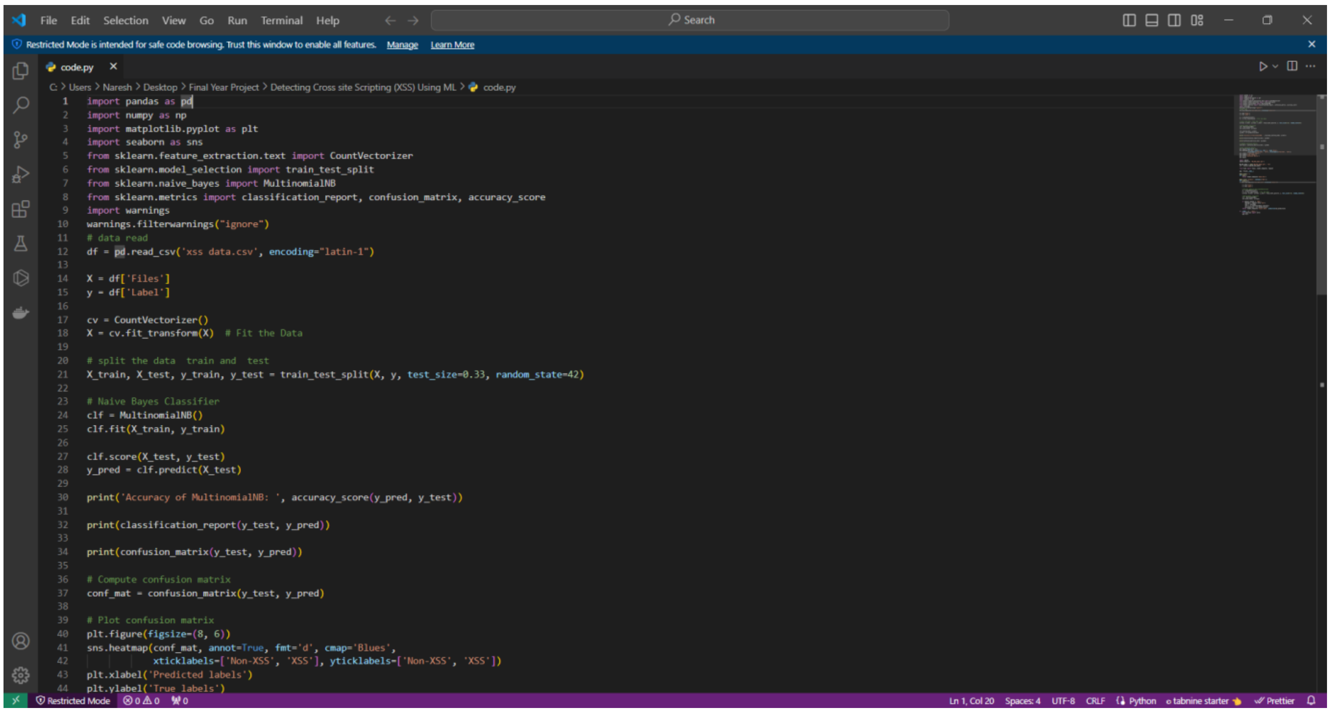
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
warnings.filterwarnings("ignore")
# data read
df = pd.read_csv('xss data.csv', encoding="latin-1")

X = df['Files']
y = df['Label']

cv = CountVectorizer()
X = cv.fit_transform(X)  # Fit the Data

# split the data  train and  test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Naive Bayes Classifier
clf = MultinomialNB()
clf.fit(X_train, y_train)

clf.score(X_test, y_test)
y_pred = clf.predict(X_test)

print('Accuracy of MultinomialNB: ', accuracy_score(y_pred, y_test))

print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

# Compute confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-XSS', 'XSS'], yticklabels=['Non-XSS', 'XSS'])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
```

Fig.8.2.1 Python Code - 1



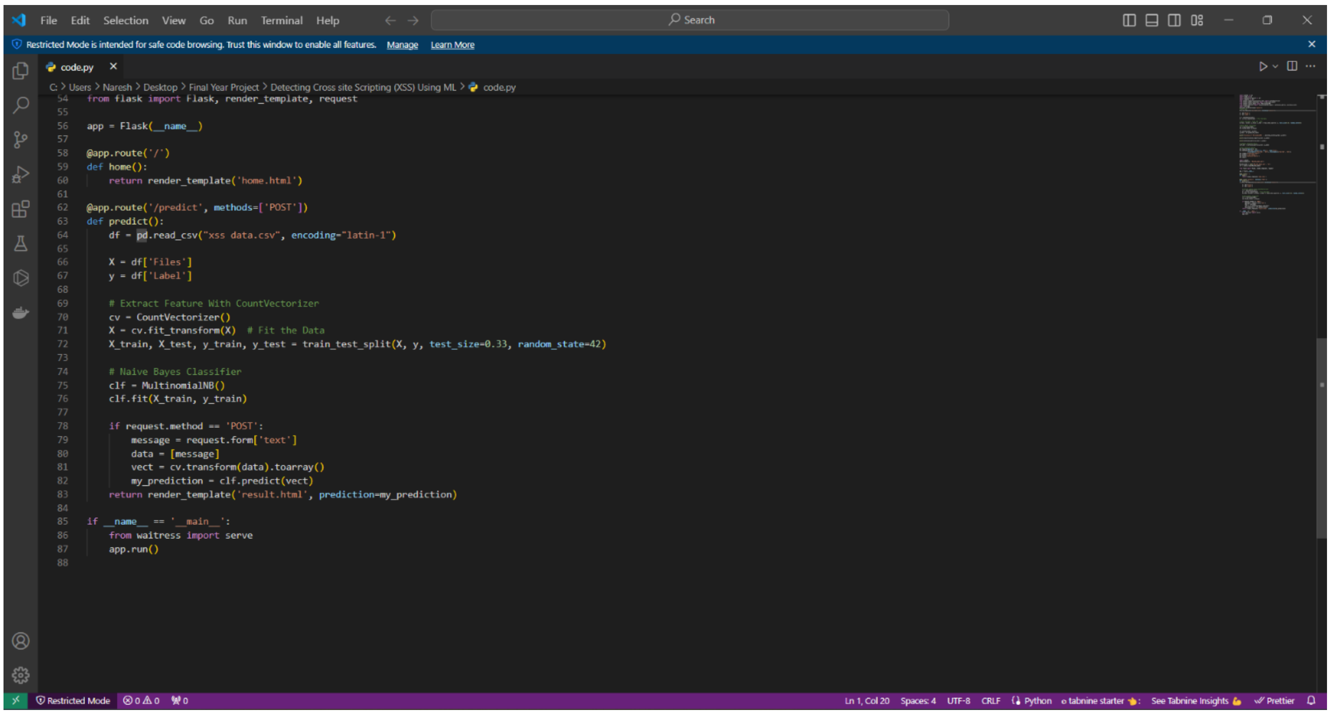
```python
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/predict', methods=['POST'])
def predict():
    df = pd.read_csv("xss data.csv", encoding="latin-1")

    X = df['Files']
    y = df['Label']

    # Extract Feature With CountVectorizer
    cv = CountVectorizer()
    X = cv.fit_transform(X)  # Fit the Data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

    # Naive Bayes Classifier
    clf = MultinomialNB()
    clf.fit(X_train, y_train)

    if request.method == 'POST':
        message = request.form['text']
        data = [message]
        vect = cv.transform(data).toarray()
        my_prediction = clf.predict(vect)
    return render_template('result.html', prediction=my_prediction)

if __name__ == '__main__':
    from waitress import serve
    app.run()
```

Fig.8.2.2 Python Code - 2



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
warnings.filterwarnings("ignore")
# data read
df = pd.read_csv('xss data.csv', encoding="latin-1")

X = df['Files']
y = df['Label']

cv = CountVectorizer()
X = cv.fit_transform(X)  # Fit the Data

# split the data  train and  test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Naive Bayes Classifier
clf = MultinomialNB()
clf.fit(X_train, y_train)

clf.score(X_test, y_test)
y_pred = clf.predict(X_test)

print('Accuracy of MultinomialNB: ', accuracy_score(y_pred, y_test))

print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

# Compute confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-XSS', 'XSS'], yticklabels=['Non-XSS', 'XSS'])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
```

Fig.8.2.3 Python Code - 3



```python
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/predict', methods=['POST'])
def predict():
    df = pd.read_csv("xss data.csv", encoding="latin-1")

    X = df['Files']
    y = df['Label']

    # Extract Feature With CountVectorizer
    cv = CountVectorizer()
    X = cv.fit_transform(X)  # Fit the Data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

    # Naive Bayes Classifier
    clf = MultinomialNB()
    clf.fit(X_train, y_train)

    if request.method == 'POST':
        message = request.form['text']
        data = [message]
        vect = cv.transform(data).toarray()
        my_prediction = clf.predict(vect)
    return render_template('result.html', prediction=my_prediction)

if __name__ == '__main__':
    from waitress import serve
    app.run()

```

Fig.8.2.4 Python Code - 4

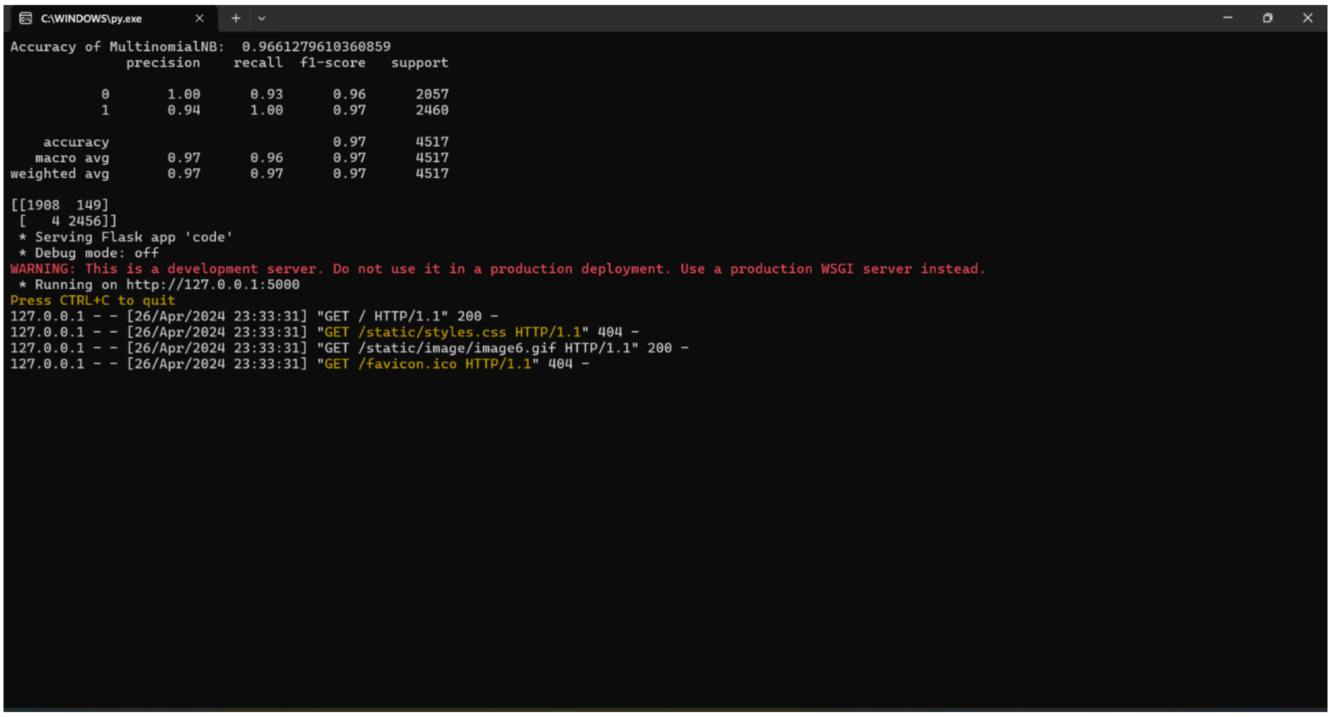

```
Accuracy of MultinomialNB:  0.9661279610360859
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      2057
           1       0.94      1.00      0.97      2460

    accuracy                           0.97      4517
   macro avg       0.97      0.96      0.97      4517
weighted avg       0.97      0.97      0.97      4517

[[1908  149]
 [   4 2456]]
 * Serving Flask app 'code'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [26/Apr/2024 23:33:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Apr/2024 23:33:31] "GET /static/styles.css HTTP/1.1" 404 -
127.0.0.1 - - [26/Apr/2024 23:33:31] "GET /static/image/image6.gif HTTP/1.1" 200 -
127.0.0.1 - - [26/Apr/2024 23:33:31] "GET /favicon.ico HTTP/1.1" 404 -
```

Fig.8.2.5 Output Console

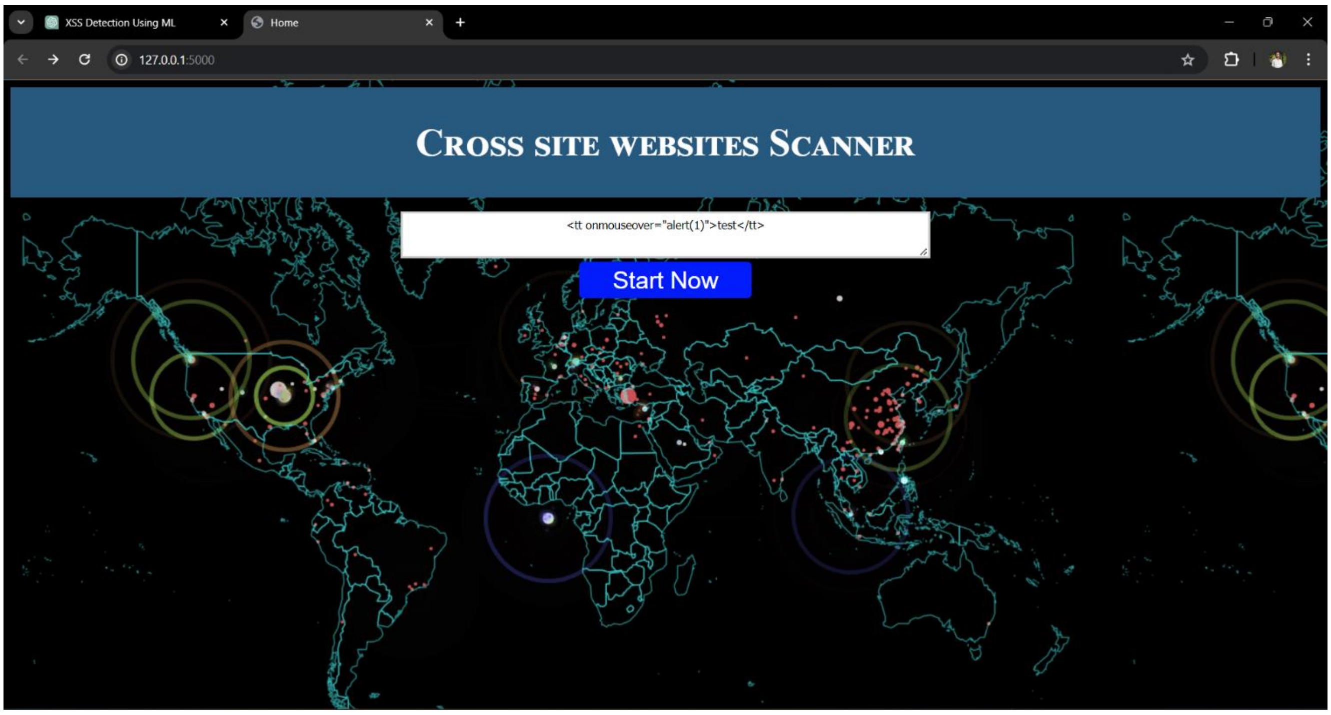
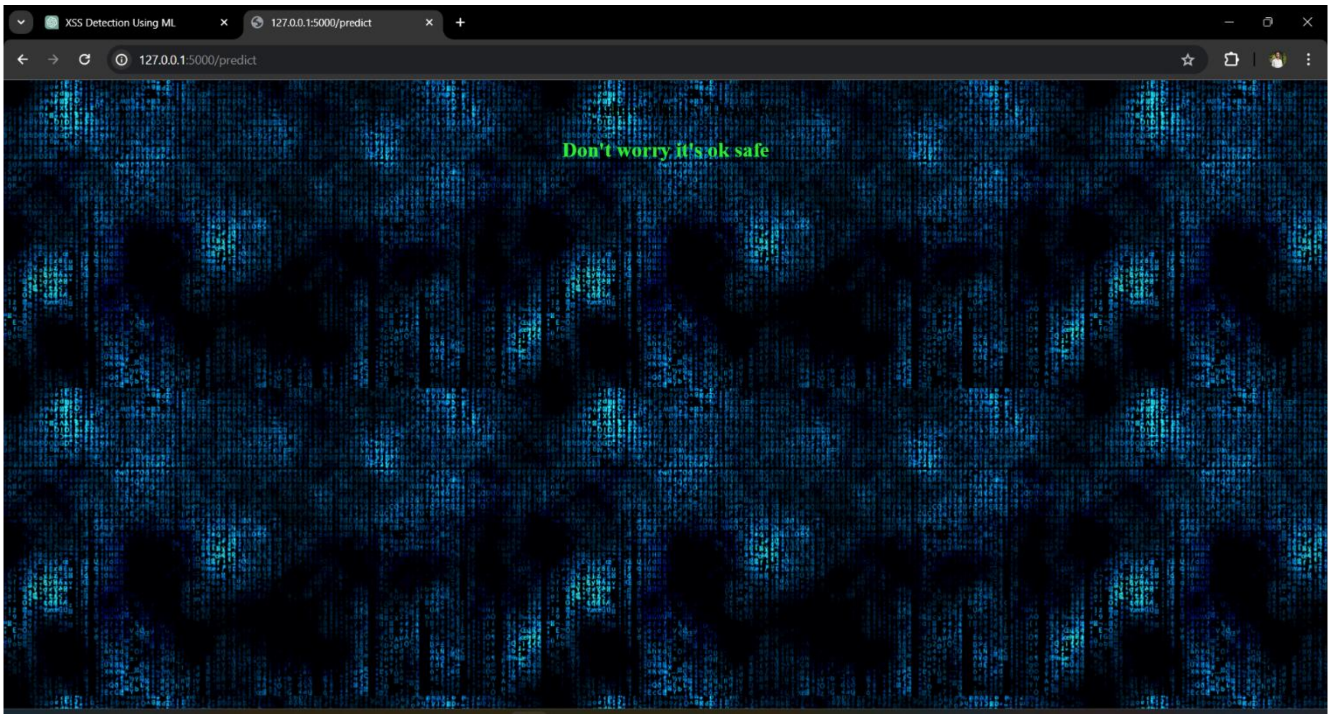
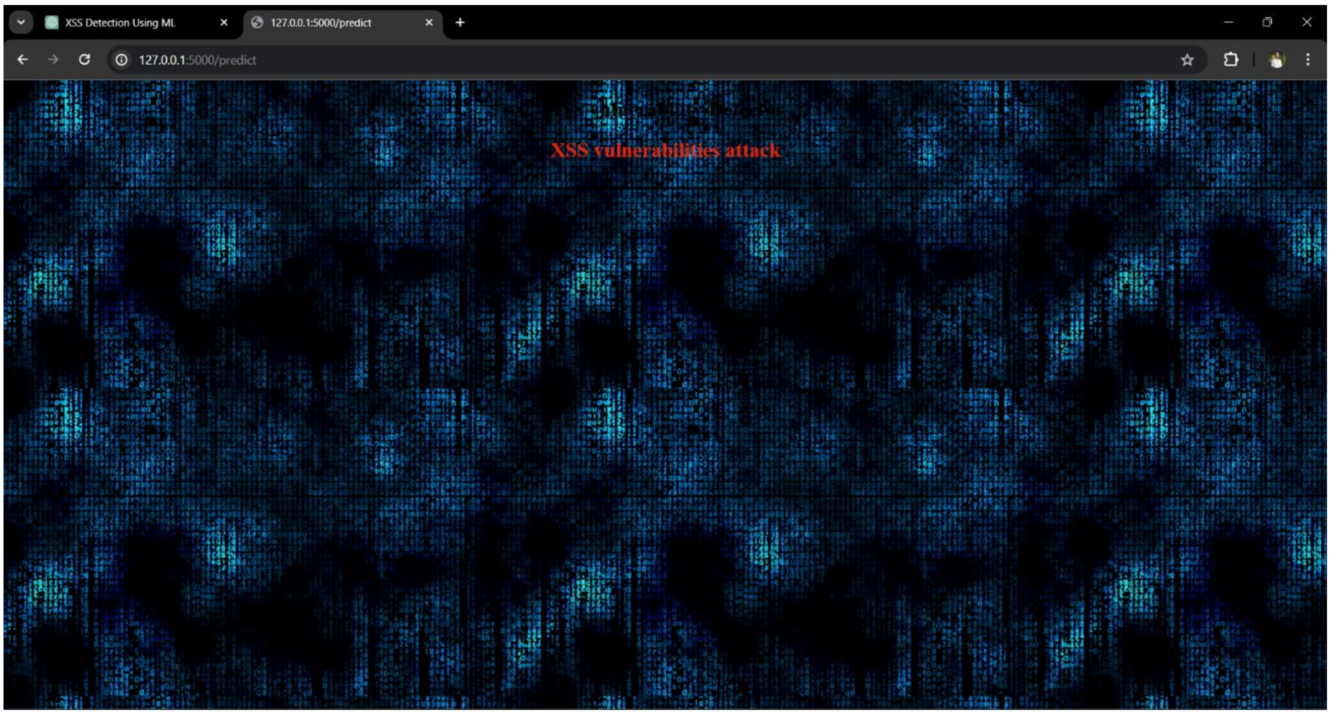

XSS Detection Using ML
Home
127.0.0.1:5000
Cross site websites Scanner
<tt onmouseover="alert(1)">test</tt>
Start Now

Fig.8.2.6 Web Application XSS



Don't worry it's ok safe

Fig.8.2.7 Web Application Result



XSS vulnerabilities attack

Fig.8.2.8 Web Application Result

9. CONCLUSION & FUTURE WORK

9.1 CONCLUSION:

Cross-site scripting (XSS) is a significant security issue for web applications, and machine learning techniques can be used to detect and prevent such attacks. Multinomial Naive Bayes (MNB) with Count Vectorizer is one such approach that can be used for XSS detection.

Count Vectorizer is a feature extraction technique that transforms text data into numerical vectors, which can then be used as input to machine learning algorithms. MNB with Count Vectorizer uses the frequency counts of words in the text as features, and the algorithm can be trained on a dataset of known malicious and benign web pages to classify new web pages as either malicious or benign.

The advantage of using MNB with Count Vectorizer for XSS detection is that it is computationally efficient and can handle large datasets with many features. It also performs well with high-dimensional data and can identify important features that contribute to the classification task. Additionally, this approach can be used to analyze the content of web pages without requiring knowledge of the underlying web application code.

However, the performance of MNB with Count Vectorizer for XSS detection may be affected by the quality and size of the training dataset and the effectiveness of the feature extraction technique. In addition, this approach may not be able to detect more sophisticated XSS attacks that use advanced evasion techniques. In conclusion, MNB with Count Vectorizer is a promising approach for XSS detection, and it can provide efficient and effective protection for web applications. However, it is important to consider the limitations of this approach and to continue research to improve its performance and address its weaknesses.

MNB with Count Vectorizer offers an efficient and effective approach for XSS detection by leveraging machine learning techniques to analyze the textual content of web pages. Its computational efficiency and ability to handle large datasets make it suitable for real-world applications. However, its performance may vary depending on the quality and size of the training dataset, and it may not detect more sophisticated XSS attacks. Continued research and refinement are necessary to enhance its effectiveness and address its limitations, ultimately bolstering the security posture of web applications against XSS threats.

9.2 FUTURE WORK:

As technology evolves and cyber threats become increasingly sophisticated, the field of XSS detection continues to demand innovation and advancement. Future work in this area should focus on addressing existing challenges and exploring new avenues to enhance the effectiveness and efficiency of XSS detection techniques. Here are several potential areas for future research and development:

Advanced Machine Learning Techniques: While Multinomial Naive Bayes (MNB) with Count Vectorizer shows promise for XSS detection, exploring more advanced machine learning algorithms such as deep learning and ensemble methods could lead to improved accuracy and robustness. These techniques have the potential to capture complex patterns and relationships in web page content, thereby enhancing the detection of subtle XSS attacks.

Feature Engineering and Selection: Investigating novel feature extraction techniques tailored specifically for XSS detection could enhance the discriminatory power of machine learning models. This includes exploring domain-specific features, syntactic and semantic analysis, and incorporating contextual information to better capture the characteristics of malicious scripts.

Adversarial Attack Detection: Developing mechanisms to detect and mitigate adversarial attacks aimed at evading XSS detection systems is crucial. This involves studying adversarial techniques used by attackers to bypass detection mechanisms and designing resilient algorithms capable of identifying and thwarting such attacks.

Behavioral Analysis: Integrating behavioral analysis techniques into XSS detection systems could provide additional layers of defense against evolving attack strategies. By monitoring and analyzing user interactions with web applications in real-time, anomalous behavior indicative of XSS attacks can be identified and mitigated proactively.

Hybrid Approaches: Investigating hybrid approaches that combine machine learning techniques with rule-based systems and signature-based detection methods could offer synergistic benefits. By leveraging the strengths of each approach, hybrid systems can achieve higher detection rates while minimizing false positives and false negatives.

Scalability and Efficiency: As web applications continue to grow in complexity and scale, ensuring the scalability and efficiency of XSS detection systems becomes paramount. Future work should focus on developing lightweight and scalable algorithms capable of processing large volumes of data in real-time without sacrificing performance.

10. REFERENCES

[1] *M. Khan, A. R. Díaz-Uriarte, 2023. "Feature Engineering for Effective XSS Detection with Machine Learning".*

[2] *M. F. Kaaniche, H. J. Rebeiz, M. Kaaniche, 2022. "XSS-Guard: Precise Dynamic Prevention of Cross-Site Scripting Attacks".*

[3] *A. S. Patil, A. B. Jadhav, 2021 ." A Survey of Machine Learning Techniques for XSS Vulnerability Detection."*

[4] *Y. Liu, W. Dai, L. Qiao, 2021."Deep-XSS: Cross-Site Scripting Detection with Neural Networks"*

[5] *L. Huang, M. H. Nguyen, H. Y. Kim, J. J. Ahn, 2021."XSS-Detect: An Intelligent XSS Detection System Based on Machine Learning".*

[6] *Hsing-Chung Chen, Pi-Hsien Chang, 2021."Detection and Prevention of Cross-site Scripting Attack with Combined Approaches".*

[7] *Aldahdooh A, Hamidouche W, Fezza SA, Déforges O (2022) Adversarial example detection for DNN models: a review and experimental comparison. Artif Intell Rev 2022:1–60.*

[8] *Abdullah Alqarni A, Alsharif N, Ahmad Khan N, Georgieva L, Pardade E, Alzahrani YM (2022) MNNXSS: modular neural network based approach for XSS attack detection. Comput Mater Continua 70(2):4075–4085*

[9] *Banerjee, R., Baksi, A., Singh, N., & Bishnu, S. K. (2020, October 2). "Detection of XSS in web applications using Machine Learning Classifers."*

[10] *Brewer R (2016) Ransomware attacks: detection, prevention and cure. Netw Secur 2016(9).*

[11] *Chauhan S, Vig L, Filippo De Grazia M, Corbetta M, Ahmad S, Zorzi M (2019) A Comparison of shallow and deep learning methods for predicting cognitive performance of stroke patients from mri lesion images. Front Neuroinform.*

[12] *Chen T, Liu J, Xiang Y, Niu W, Tong E, Han Z (2019) Adversarial attack and defense in reinforcement learning-from AI security view. Cybersecurity 2:1.*

[13] *Cimpanu Catalin. (2018). British Airways breach caused by the same group that hit Ticketmaster | ZDNet. ZDNET, A RED VENTURES COMPANY.*

[14] *Conti, M., Dargahi, T., & Dehghantanha, A. (2018). Cyber threat intelligence: Challenges and opportunities. In Advances in Information Security (Vol. 70, pp. 1–6). Springer New York LLC.*

[15] *Abaimov S, Bianchi G (2019) CODDLE: Code-injection detection with deep learning. IEEE Access 7:128617–128627.*

Detection of Cross-Site Scripting (XSS) Using Machine Learning Methods

Dr.M.Duraipandian[1], Mr.S.Narasimmaraj[2], Mr.K.ThanushKumar[3], Mr.D.K.Vignesh[4], Mr.S.Yuvaraj[5].

[1]Head of the Department, Department of Information Technology,Hindusthan Institute of Technology, Coimbatore,Tamilnadu.

[2,3,4,5]Student,Department of Information Technology,Hindusthan Institute of Technology, Coimbatore,Tamilnadu.

Abstract: This paper addresses the pressing issue of cross-site scripting (XSS) attacks, which rank among the top web application risks according to OWASP. Historically underestimated, XSS poses significant threats, potentially compromising sensitive data like payment information or session tokens. Leveraging machine learning, specifically the Multinomial Naive Bayes classifier, our study presents a novel approach for detecting XSS attacks. Through experimental validation, we demonstrate the efficacy of our model in mitigating XSS vulnerabilities, thereby enhancing web security measures.

Keywords: Cross-Site Scripting (XSS), Machine Learning, Multinomial Naive Bayes Classifier, Web Application Security, OWASP, etc.,

1. INTRODUCTION

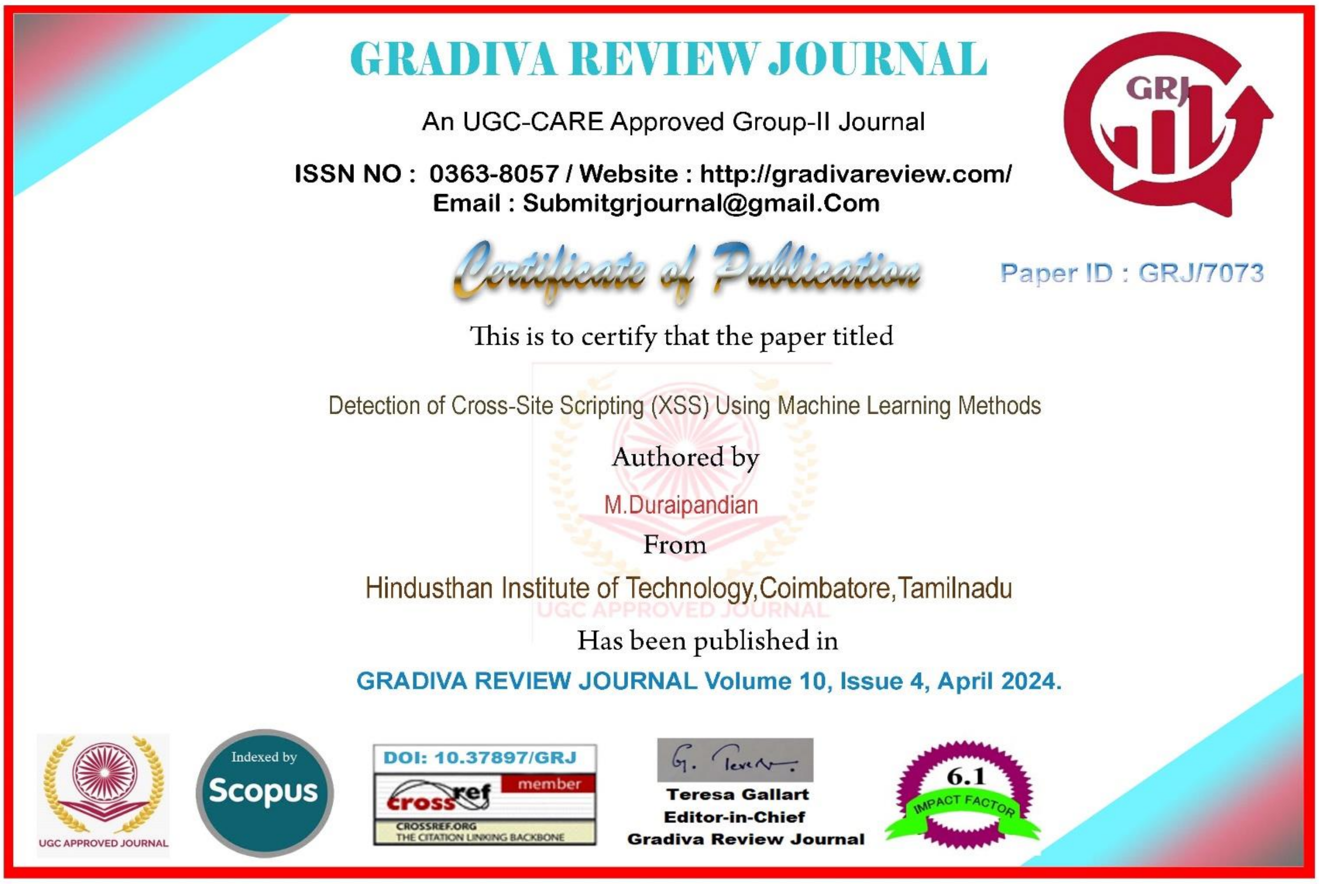
In the digital age, web applications serve as fundamental conduits for storing and processing sensitive data, rendering them prime targets for malicious exploitation. Among the plethora of cyber threats, cross-site scripting (XSS) stands out as a formidable adversary. XSS, characterized by the injection of malicious code into web pages, engenders a perilous nexus between users and attackers' servers.

By exploiting vulnerabilities within web servers or users' devices, XSS facilitates unauthorized access, leveraging authenticated user privileges for data theft or unauthorized operations. Despite its historical trivialization, XSS poses multifaceted risks, including cookie hijacking, webpage manipulation, and dynamic content injection, underscoring the exigency of robust security measures for web applications.

The prevalence of XSS vulnerabilities, stemming primarily from deficiencies in client input validation, is starkly highlighted by its persistent inclusion in OWASP's top ten threats. This paper endeavors to elucidate the evolving threat landscape of XSS, dispelling misconceptions and advocating for proactive mitigation strategies to fortify the security posture of web applications in an increasingly hostile cyber environment.

2. Why Machine Learning?

Machine learning applications have permeated various sectors, revolutionizing industries with their ability to extract insights from vast volumes of data. From self-driving



GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : http://gradivareview.com/
Email : Submitgrjournal@gmail.Com

Certificate of Publication

Paper ID : GRJ/7073

This is to certify that the paper titled

Detection of Cross-Site Scripting (XSS) Using Machine Learning Methods

Authored by

M.Duraipandian

From

Hindusthan Institute of Technology,Coimbatore,Tamilnadu

Has been published in

GRADIVA REVIEW JOURNAL Volume 10, Issue 4, April 2024.

UGC APPROVED JOURNAL

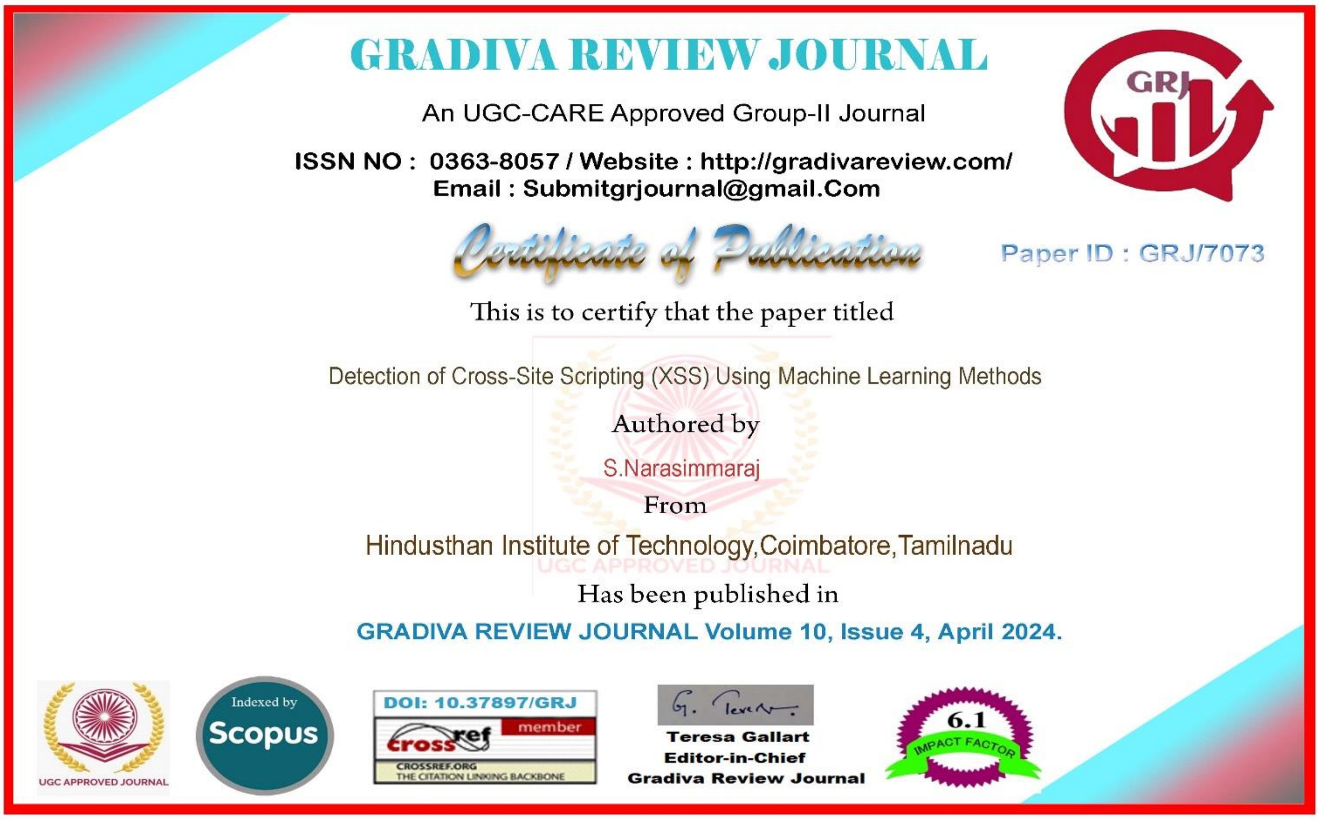
Indexed by Scopus

DOI: 10.37897/GRJ

[illegible]

Teresa Gallart
Editor-in-Chief
Gradiva Review Journal

6.1 IMPACT FACTOR



GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : http://gradivareview.com/
Email : Submitgrjournal@gmail.Com

Certificate of Publication

Paper ID : GRJ/7073

This is to certify that the paper titled

Detection of Cross-Site Scripting (XSS) Using Machine Learning Methods

Authored by

S.Narasimmaraj

From

Hindusthan Institute of Technology,Coimbatore,Tamilnadu

Has been published in

GRADIVA REVIEW JOURNAL Volume 10, Issue 4, April 2024.

UGC APPROVED JOURNAL

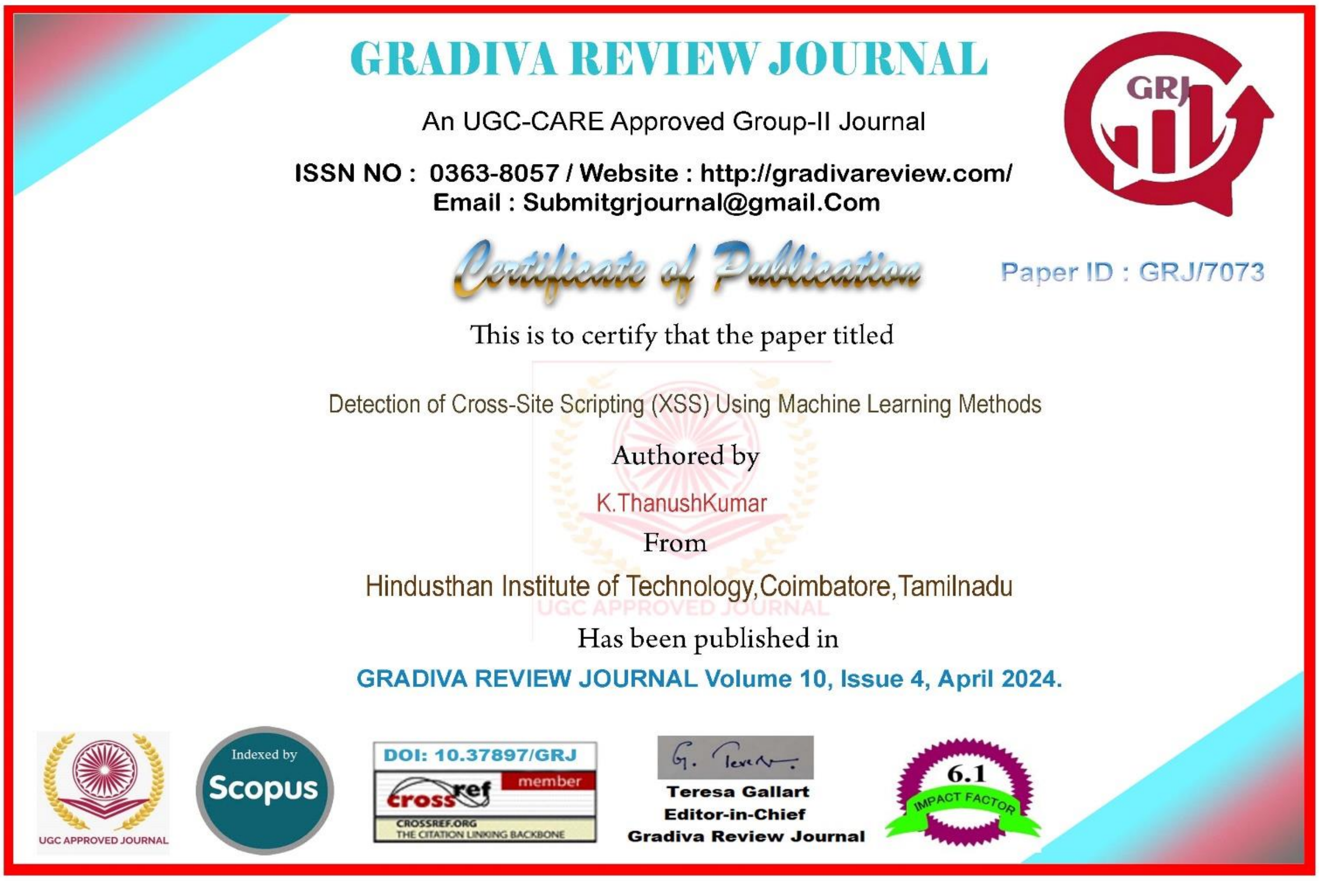
Indexed by Scopus

DOI: 10.37897/GRJ

[illegible]

Teresa Gallart
Editor-in-Chief
Gradiva Review Journal

6.1 IMPACT FACTOR


GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : http://gradivareview.com/
Email : Submitgrjournal@gmail.Com

Certificate of Publication

Paper ID : GRJ/7073

This is to certify that the paper titled

Detection of Cross-Site Scripting (XSS) Using Machine Learning Methods

Authored by

K.ThanushKumar

From

Hindusthan Institute of Technology,Coimbatore,Tamilnadu

Has been published in

GRADIVA REVIEW JOURNAL Volume 10, Issue 4, April 2024.

UGC APPROVED JOURNAL

Indexed by Scopus

DOI: 10.37897/GRJ

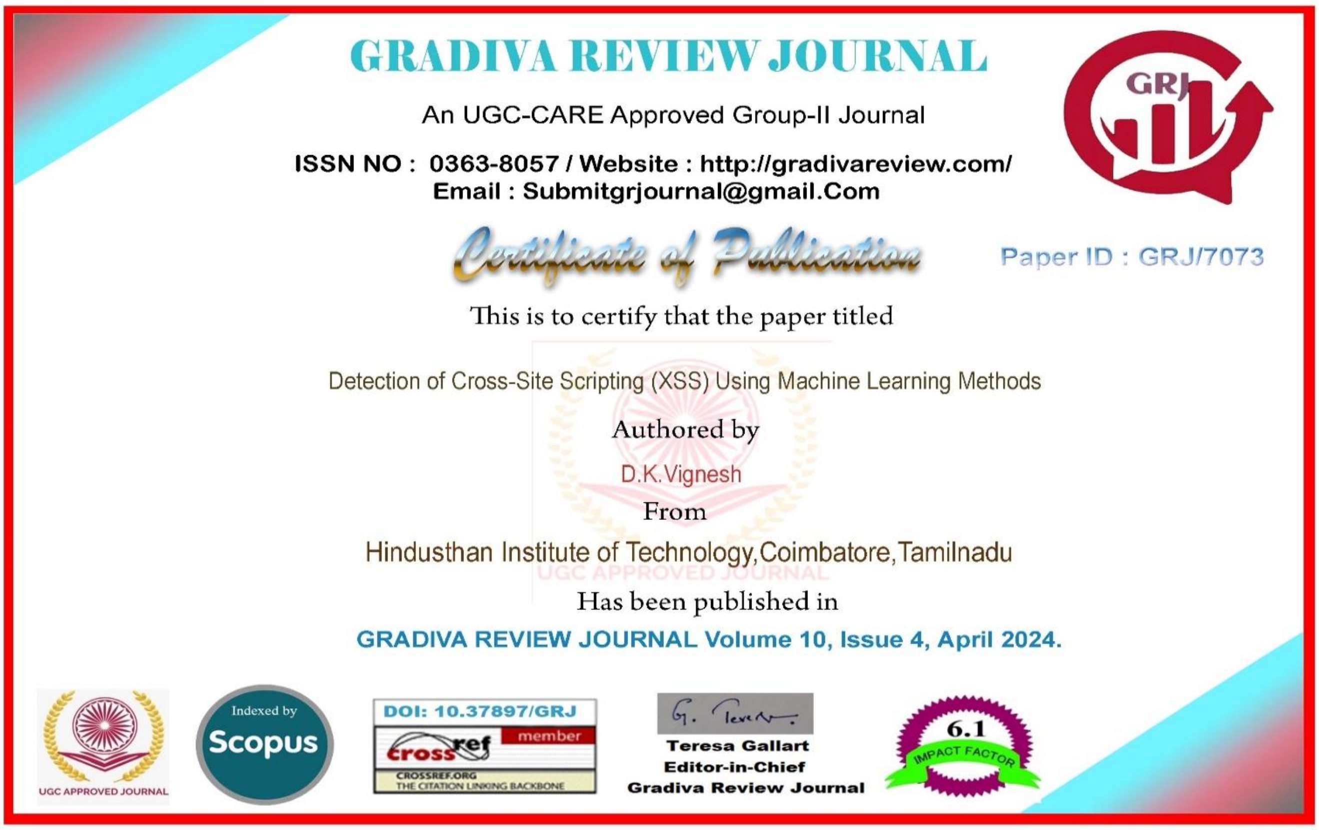
crossref member

CROSSREF.ORG THE CITATION LINKING BACKBONE

[illegible]

Teresa Gallart
Editor-in-Chief
Gradiva Review Journal

6.1 IMPACT FACTOR



GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : http://gradivareview.com/
Email : Submitgrjournal@gmail.Com

Certificate of Publication

Paper ID : GRJ/7073

This is to certify that the paper titled

Detection of Cross-Site Scripting (XSS) Using Machine Learning Methods

Authored by

D.K.Vignesh

From

Hindusthan Institute of Technology,Coimbatore,Tamilnadu

Has been published in

GRADIVA REVIEW JOURNAL Volume 10, Issue 4, April 2024.

UGC APPROVED JOURNAL

Indexed by Scopus

DOI: 10.37897/GRJ

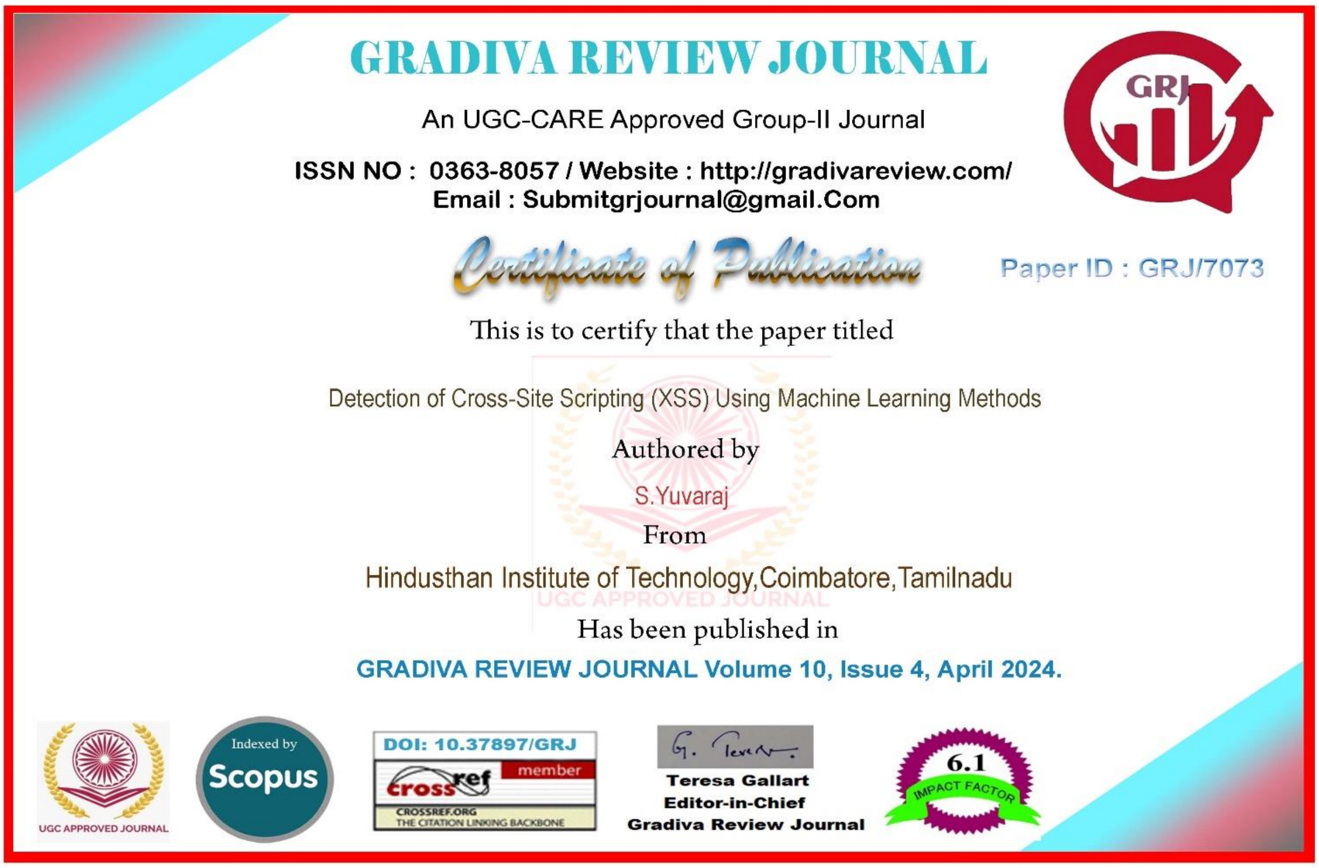
crossref member

CROSSREF.ORG THE CITATION LINKING BACKBONE

[illegible]

Teresa Gallart
Editor-in-Chief
Gradiva Review Journal

6.1 IMPACT FACTOR



GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : http://gradivareview.com/
Email : Submitgrjournal@gmail.Com

Certificate of Publication

Paper ID : GRJ/7073

This is to certify that the paper titled

Detection of Cross-Site Scripting (XSS) Using Machine Learning Methods

Authored by

S.Yuvaraj

From

Hindusthan Institute of Technology,Coimbatore,Tamilnadu

Has been published in

GRADIVA REVIEW JOURNAL Volume 10, Issue 4, April 2024.

UGC APPROVED JOURNAL

Indexed by Scopus

DOI: 10.37897/GRJ

crossref member
CROSSREF.ORG
THE CITATION LINKING BACKBONE

[illegible]
Teresa Gallart
Editor-in-Chief
Gradiva Review Journal

6.1 IMPACT FACTOR