MULTILABEL CAPEC CLASSIFICATION OF WEB ATTACKS

Enroll Numbers: 21103218, 211104032, 21104023
Name : Aibad Khan, Akshat Agrawal, Pritpal Singh
Name of Supervisor: Dr. Sangeeta Mittal

November - 2023

Submitted in Partial Fulfillment of Degree of
Bachelor of Technology in
Information Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

(I)

TABLE OF CONTENTS

Chapter No.	Topics	Page No.
Chapter - 1	**Introduction**	**9-11**
1.1.	General Introduction	
1.2.	Problem statement	
1.3.	Significance/Novelty of the Problem	
1.4.	Empirical Study	
1.5.	Brief Description of the Solution Approach	
1.6.	Comparison of existing approaches to the problem-framed	
Chapter - 2	**Literature Survey**	**12**
2.1.	Summary of papers studied	
2.2	Integrated summary of the literature studied	
Chapter - 3	**Requirement Analysis and Solution Approach**	**13-19**
3.1	Overall description of the project	
3.2	Requirement Analysis	
3.3	Solution Approach	
Chapter - 4	**Modelling and Implementation Details**	**20-26**
4.1	Design Diagrams	
	4.1.2 Use Case Diagrams	
	4.1.2 Class Diagrams / Control Flow Diagrams	
	4.1.3 Sequence Diagrams / Activity Diagrams	
4.2	Implementation details and issues	
4.3	Risk Analysis and Mitigation	
Chapter - 5	**Testing**	**27-29**
5.1	Testing Plan	
5.2	Content decomposition and type of testing required	
5.3	Test cases	
5.4	Error and Exception Handling	
5.5	Limitations of the solution	
Chapter - 6	**Findings, Conclusions, and Future Work**	**30-34**
6.1	Findings	
6.2	Conclusions	
6.3	Future Work	
References		**35**

(II)

DECLARATION

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Place:	Signature:
Date:	Name:
	Enrollment No.:

Place:	Signature:
Date:	Name:
	Enrollment No.:

Place:	Signature:
Date:	Name:
	Enrollment No.:

(III)
CERTIFICATE

This is to certify that work titled "**MULTILABEL CAPEC CLASSIFICATION OF WEB ATTACKS"** submitted by **"Aibad Khan(21103218), Pritpal Singh(21104023) and Akshat Agrawal(21104032)"** in partial fulfillment for the award of degree of **B. Tech** of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other university or institute for the award of this or any other degree or diploma.

Signature of Supervisor	…………………………………..
Name of Supervisor	…………………………………..
Designation	…………………………………..
Date	…………………………………..

(IV)
ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who has contributed to the successful completion of our Semester V minor project, titled "**MULTILABEL CAPEC CLASSIFICATION OF WEB ATTACKS"**

First and foremost, We extend our heartfelt thanks to **Dr. Vikas Saxena**, Head of the Department - CSE & IT, for his unwavering support and invaluable insights throughout the course of this project. His guidance has been instrumental in shaping our ideas and pushing us towards excellence.

We would also like to extend our sincere appreciation to our esteemed faculty coordinator, **Dr. Sangeeta Mittal**, for her exceptional guidance and mentorship. Her expertise in the field of machine learning and cybersecurity has been a constant source of inspiration. Dr. Mittal's insightful feedback and constructive criticism have played a pivotal role in refining our approach and methodology.

We are also grateful to the entire faculty of the department for their continuous support and encouragement. Their expertise and commitment to academic excellence have been a source of motivation for all of us.

Lastly, We would like to express our heartfelt thanks to our student mentor Mr. Dhruv Taneja for their dedication and hard work throughout this project. Our collaborative efforts have been crucial in the successful implementation of our machine learning model.

This project has been a significant learning experience, and we are thankful to everyone who has been a part of this journey.

(V)
SUMMARY

This machine learning (ML) project focuses on the identification and classification of web attacks by leveraging advanced multilabel machine learning algorithms. The primary objective is to categorize web attacks into Common Attack Pattern Enumeration and Classification (CAPEC) categories, providing a comprehensive understanding of the threats posed to web applications.

The project employs a diverse set of multilabel machine learning algorithms, including but not limited to Binary Relevance, Random Forest, Label Powerset, Multi-Output Classifier, and LightGBM. These algorithms are selected for their ability to handle the complex and interconnected nature of web attack patterns, allowing for a more nuanced and accurate classification of threats.

The use of Binary Relevance facilitates the handling of multiple CAPEC categories simultaneously, while Random Forest provides an ensemble learning approach for improved predictive performance. The Label Powerset method considers all possible label combinations, offering a comprehensive approach to multilabel classification. The Multi-Output Classifier further enhances the model's capability to assign multiple CAPEC categories to a given web attack, capturing the intricate relationships between different attack patterns. Lastly, the utilization of LightGBM, a gradient boosting framework, contributes to faster and more efficient training of the machine learning models.

By implementing these multilabel machine learning algorithms, the project aims to enhance the accuracy and efficiency of web attack classification, enabling organizations to proactively identify and mitigate diverse threats to their web applications. This comprehensive approach to CAPEC category classification provides a valuable tool for cybersecurity professionals to strengthen their defenses against evolving web-based attacks.

(VI)

LIST OF TABLES

Table No.	Title	Page No.
1	Integrated summary of literature studied	12
2	No. of different CAPEC classification assigned to a web request	15
3	Final set of selected features and tables	16
4	Summary of metrics by algorithm and model, ordered by recall score	32

(VII)
LIST OF FIGURES

Fig no.	Figure Title	Page No.
1	Increase in the no. of attacks according to years (2011 - 2018)	9
2	SRBH - 2020 Dataset	13
3	Statistical information about the dataset	14
4	Control flow diagram	20
5	Sequence diagram	20
6	Dropping irrelevant columns	21
7	Removing redundant data	21
8	Count encoding	22
9.1 - 9.6	Implementation of ML algorithms for multilabel classification	23-24
10	Creating a pipeline	24
11.1 - 11.3	Testing user inputs	27-28
12	Hamming loss for various algorithms	33
13	Metrics values by algorithm/model combination	33

Chapter - 1 Introduction

1.1 General Introduction

Every year there is a significant increase in the number of attacks against web servers and applications, e-commerce platforms, financial and government institutions, large corporations, etc. are targeted by web attacks for economic or ideological reasons. According to Cisco (Cisco, 2018), 14.5 million DDoS attacks are expected in 2022. Also, SQL Injection (SQLi) and Cross-site Scripting (XSS) attacks are easy and powerful methods for attacking a website (Johari and Sharma, 2012). The impact of cyber-attacks suffered by companies threatened their viability in 17% of cases, reported specialist insurer Hiscox (Hiscox, 2021), with their website becoming the first point of entry in 29% of cases.(1)

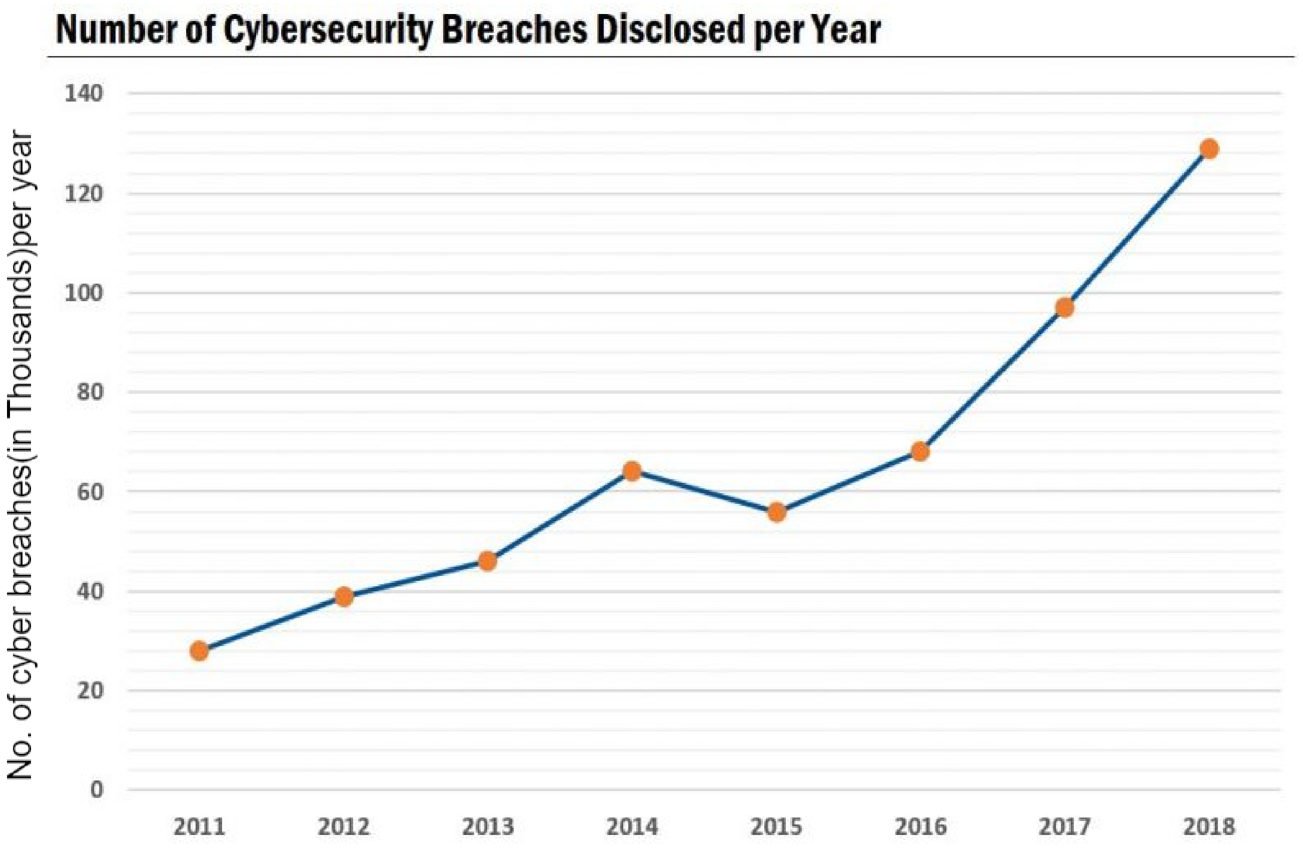


Fig1. Increase in the Number of Internet attacks according to Years (2011- 2018)

Source: https://blog.auditanalytics.com/trends-in-cybersecurity-breaches-continue-in-2019/ as seen on 20 Nov 2023

1.2 Problem statement

To build a model for identifying the web attacks and classify them under the CAPEC catalog by using MultiLabel Classification.

1.3 Significance/Novelty of the Problem

Significance:

Identifying and classifying different types of attacks, such as SQL injection, path traversal, and protocol manipulation, is crucial for developing effective defense mechanisms. Understanding the specific nature of an attack allows for the implementation of targeted security measures to mitigate the threat. Classification helps in identifying vulnerabilities in software systems. By recognizing patterns associated with different attacks, organizations can proactively address potential weaknesses in their applications. Effective classification contributes to swift incident response. When an attack is detected, knowing its type allows security teams to respond appropriately and apply the necessary countermeasures.The ongoing analysis and classification of attacks contribute to a feedback loop for improving security systems. This continuous improvement is vital in the ever-evolving landscape of cybersecurity. Cybersecurity incidents can have severe economic consequences. The cost of data breaches, downtime, and reputation damage can be significant. Effective attack classification contributes to minimizing these economic impacts.

Novelty

The novelty in classifying attacks through multi-label classification lies in the ability to discern and categorize complex, multifaceted cyber threats with a granular level of precision. This approach goes beyond traditional binary classifications, offering a nuanced understanding of simultaneous attack vectors. By utilizing multi-label classification, security systems can adapt to the sophisticated nature of modern cyber threats, accurately identifying and labeling various attack types in a single instance. This innovation enables a more proactive and dynamic defense strategy, empowering organizations to stay ahead of evolving threats and fortify their cybersecurity measures against an ever-changing landscape of potential risks.

1.4 Empirical Study

In this study, a diverse set of machine learning algorithms, such as decision trees, random forests,ensemble methods are employed to build and train models on the dataset. Each algorithmic approach is carefully configured, and hyperparameter tuning is performed to optimize their performance. The study assesses the models' accuracy, precision, recall, and F1-score metrics to comprehensively evaluate their ability to correctly classify different types of web attacks.

1.5 Brief Description of the Solution Approach

The solution is a comprehensive methodology tailored to tackle escalating web attack challenges. It starts with careful feature selection, emphasizing crucial attributes for attack identification and classification. Using count encoding enhances the model's ability to spot patterns in categorical data. Diverse machine learning models like Binary Relevance, Random Forest, Multi Output Classifier, Adaboost, LightGBM, Classifier Chain, and Labeled Power Set work together, forming a robust defense against various attack types. This ensemble aims to capitalize on each model's strengths, ensuring a more accurate defense. By integrating advanced ML techniques, the approach not only detects but also systematically classifies attacks under the CAPEC catalog, contributing to a sophisticated cybersecurity framework.

1.6 Comparison of existing approaches to the problem-framed

Web Application Firewall (WAF) as a Common Solution:

One of the primary methods for safeguarding web applications is the use of Web Application Firewalls (WAFs). These tools are positioned between the application and the user, scrutinizing incoming traffic at the application layer of the OSI model. WAFs operate by identifying attack patterns and subsequently blocking malicious traffic. They are language-independent, acting before the execution of code, and rely on a database of signatures or rules for inspection. Updating this database is crucial for keeping up with emerging threats.

Challenges and Limitations of WAFs:

Despite their widespread use, WAFs face challenges in countering various evasion techniques proposed by attackers. These techniques include Normalization Method, HTTP Parameter Pollution (HPP), HTTP Parameter Fragmentation (HPF), logical requests AND / OR, and manipulations like replacing SQL functions with synonyms or using comments and case changing. These methods typically exploit differences in how WAFs, web servers, and backend applications interpret traffic, emphasizing protocol layer vulnerabilities.

Machine Learning-Based Detection as an Enhanced Approach:

To overcome the limitations of traditional WAFs, there is a move towards integrating machine learning (ML) into web attack detection. ML-based detection offers a more adaptive and dynamic approach by monitoring applications for security anomalies, uncommon behaviors, and common web application attacks. This shift, as proposed by Auxilia and Tamilselvan in 2010, involves adopting a negative security model. Unlike traditional WAFs that rely on predefined rules and signatures, ML-based detection can better adapt to emerging threats and evolving attack patterns, providing a more robust defense mechanism against web attacks.

Chapter - 2 Literature Survey

2.1. Summary of papers studied

This paper introduces a novel approach to enhancing web attack detection tools by presenting a new multi-label dataset for classifying web attacks based on the Common Attack Pattern Enumeration and Classification (CAPEC) system. The authors propose a new method for feature extraction based on ASCII values, focusing on computing the average of the sum of ASCII values for characters in each field of a web request. The study evaluates various combinations of machine learning algorithms, specifically LightGBM and CatBoost, along with multi-label classification models.
The training and test data are sourced from the newly introduced SR-BH 2020 multi-label dataset. Results demonstrate that the average of the sum of ASCII values is effective for numeric encoding and feature extraction in web requests. The SR-BH 2020 dataset facilitates the training and evaluation of multi-label classification models, enabling the CAPEC classification of different attacks on web systems. The paper highlights the superiority of a two-phase model combined with the MultiOutputClassifier module of the scikit-learn library and the CatBoost algorithm in classifying attacks across various criticality scenarios. The experimental findings suggest that integrating machine learning algorithms with multi-phase models enhances the prediction of web attacks. Additionally, the use of a multi-label dataset is deemed suitable for training models that provide detailed information about the type of attack being experienced by a web system.

2.2 Integrated summary of the literature studied

Table 1

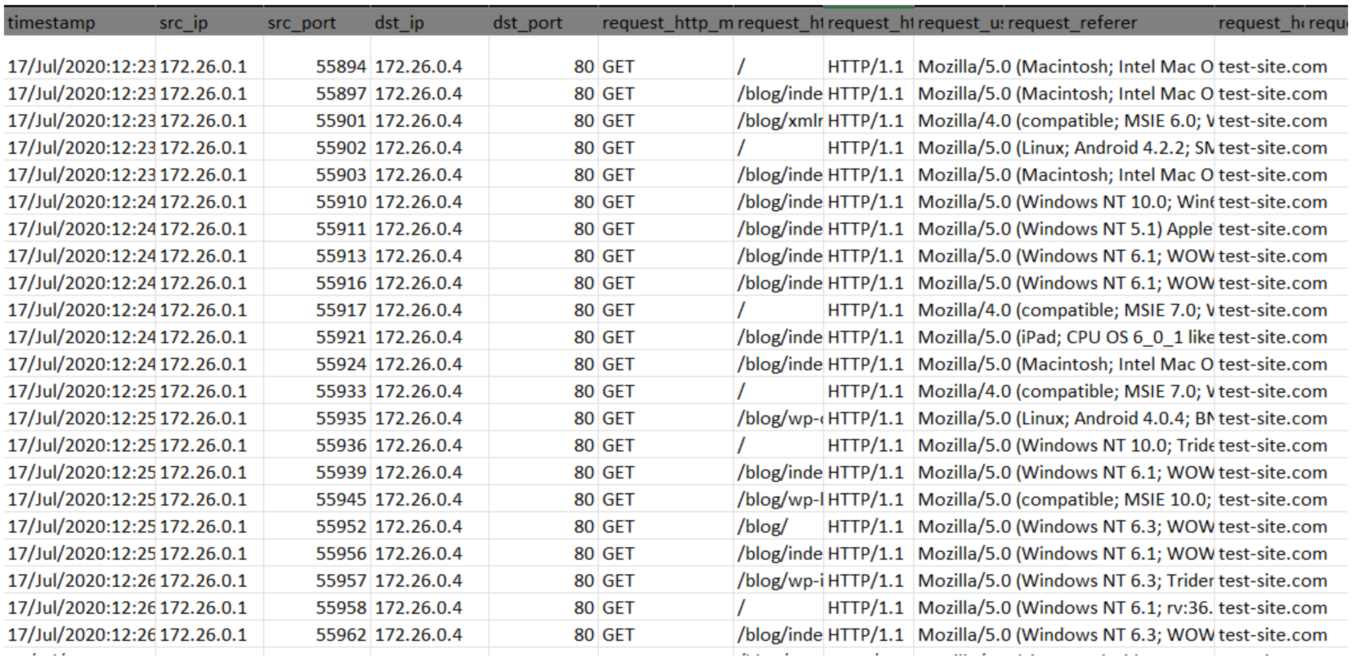
Authors	Topic	Key Characteristics
Tomás Sureda Riera, Juan-Ramón Bermejo Higuera, Juan-Antonio Sicilia Montalvo	A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques	This paper presents a new multi-label dataset for classifying web attacks based on CAPEC classification, a new way of feature extraction based on ASCII values, and the evaluation of several combinations of models and algorithms.
Qijing Qiao, Ruitao Feng, Sen Chen,	Multi-label Classification for Android Malware Based on Active Learning	In this paper, the researchers propose MLCDroid, an ML-based multi-label classification approach that can directly indicate the existence of pre-defined malicious behaviors. With an in-depth analysis, we summarize 6 basic malicious behaviors from real-world malware with security reports and construct a labeled dataset
Xin Zhang, Rabab Abdelfattah, Yuqi Song, Xiaofeng Wang	An Effective Approach for Multi-label Classification with Missing Labels.	Proposes a pseudo-label based approach to reduce the cost of annotation without bringing additional complexity to the existing classification networks.

Chapter - 3 Requirement Analysis and Solution Approach

3.1 Overall description of the project

1. The primary objective of this project is to develop an advanced multilabel classification model for effectively categorizing web traffic into distinct classes, including 'normal,' 'protocol manipulation,' 'SQL injection,' and 'path traversal.'
2. The goal is to enhance the security and resilience of web applications by accurately identifying and classifying potential security threats.
3. Through meticulous feature engineering, model selection, and training, we aim to achieve a robust and proactive defense system capable of outperforming traditional approaches.
4. The project also seeks to align the classification system with the identified threat categories, contributing to a structured and comprehensive understanding of potential security risks.

3.2 Dataset description

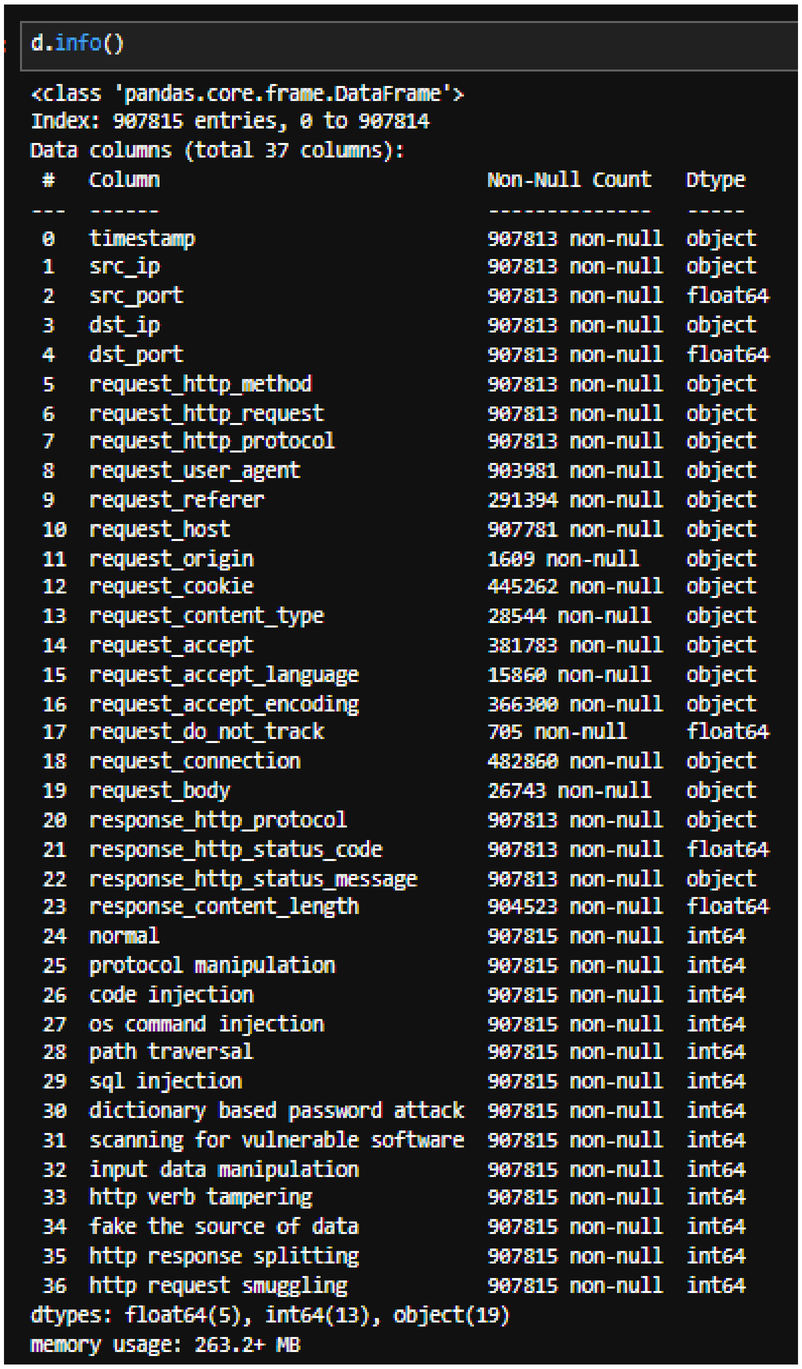


timestamp	src_ip	src_port	dst_ip	dst_port	request_http_m	request_ht	request_ht	request_us	request_referer	request_h	requ
17/Jul/2020:12:23	172.26.0.1	55894	172.26.0.4	80	GET	/	HTTP/1.1	Mozilla/5.0 (Macintosh; Intel Mac O		test-site.com	
17/Jul/2020:12:23	172.26.0.1	55897	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Macintosh; Intel Mac O		test-site.com	
17/Jul/2020:12:23	172.26.0.1	55901	172.26.0.4	80	GET	/blog/xmlr	HTTP/1.1	Mozilla/4.0 (compatible; MSIE 6.0; V		test-site.com	
17/Jul/2020:12:23	172.26.0.1	55902	172.26.0.4	80	GET	/	HTTP/1.1	Mozilla/5.0 (Linux; Android 4.2.2; SM		test-site.com	
17/Jul/2020:12:23	172.26.0.1	55903	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Macintosh; Intel Mac O		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55910	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 10.0; Win		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55911	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 5.1) Apple		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55913	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 6.1; WOW		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55916	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 6.1; WOW		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55917	172.26.0.4	80	GET	/	HTTP/1.1	Mozilla/4.0 (compatible; MSIE 7.0; V		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55921	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (iPad; CPU OS 6_0_1 like		test-site.com	
17/Jul/2020:12:24	172.26.0.1	55924	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Macintosh; Intel Mac O		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55933	172.26.0.4	80	GET	/	HTTP/1.1	Mozilla/4.0 (compatible; MSIE 7.0; V		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55935	172.26.0.4	80	GET	/blog/wp-c	HTTP/1.1	Mozilla/5.0 (Linux; Android 4.0.4; BN		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55936	172.26.0.4	80	GET	/	HTTP/1.1	Mozilla/5.0 (Windows NT 10.0; Tride		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55939	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 6.1; WOW		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55945	172.26.0.4	80	GET	/blog/wp-l	HTTP/1.1	Mozilla/5.0 (compatible; MSIE 10.0;		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55952	172.26.0.4	80	GET	/blog/	HTTP/1.1	Mozilla/5.0 (Windows NT 6.3; WOW		test-site.com	
17/Jul/2020:12:25	172.26.0.1	55956	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 6.1; WOW		test-site.com	
17/Jul/2020:12:26	172.26.0.1	55957	172.26.0.4	80	GET	/blog/wp-i	HTTP/1.1	Mozilla/5.0 (Windows NT 6.3; Trider		test-site.com	
17/Jul/2020:12:26	172.26.0.1	55958	172.26.0.4	80	GET	/	HTTP/1.1	Mozilla/5.0 (Windows NT 6.1; rv:36.		test-site.com	
17/Jul/2020:12:26	172.26.0.1	55962	172.26.0.4	80	GET	/blog/inde	HTTP/1.1	Mozilla/5.0 (Windows NT 6.3; WOW		test-site.com	

Fig2. SRBH 2020 dataset

Most datasets are composed of artificially generated traffic and, to our best knowledge, all datasets available for training and/or evaluation of machine learning models provide only labeling of the request in terms of normality or attack, without specifying in any case what type(s) of attack(s) is/are being suffered.

For this reason, one of the main achievements of this work is the generation of a new dataset (the SR-BH 2020 dataset) that collects different types of attacks, coming from real traffic data (generated by collecting real traffic in a honeypot exposed to the Internet for 12 days), with multi-labels, that report the normality of the request, or the CAPEC classification of the type or types of attack that the web request represents.This dataset, to our knowledge, is the first one that allows the training and evaluation of multi-label machine learning models and algorithms, which can provide the CAPEC classification of the attack(s) that a web application is suffering.



```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 907815 entries, 0 to 907814
Data columns (total 37 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   timestamp                       907813 non-null  object
 1   src_ip                          907813 non-null  object
 2   src_port                        907813 non-null  float64
 3   dst_ip                          907813 non-null  object
 4   dst_port                        907813 non-null  float64
 5   request_http_method             907813 non-null  object
 6   request_http_request            907813 non-null  object
 7   request_http_protocol           907813 non-null  object
 8   request_user_agent              903981 non-null  object
 9   request_referer                 291394 non-null  object
 10  request_host                    907781 non-null  object
 11  request_origin                  1609 non-null    object
 12  request_cookie                  445262 non-null  object
 13  request_content_type            28544 non-null   object
 14  request_accept                  381783 non-null  object
 15  request_accept_language         15860 non-null   object
 16  request_accept_encoding         366300 non-null  object
 17  request_do_not_track            705 non-null     float64
 18  request_connection              482860 non-null  object
 19  request_body                    26743 non-null   object
 20  response_http_protocol          907813 non-null  object
 21  response_http_status_code       907813 non-null  float64
 22  response_http_status_message    907813 non-null  object
 23  response_content_length         904523 non-null  float64
 24  normal                          907815 non-null  int64
 25  protocol manipulation           907815 non-null  int64
 26  code injection                  907815 non-null  int64
 27  os command injection            907815 non-null  int64
 28  path traversal                  907815 non-null  int64
 29  sql injection                   907815 non-null  int64
 30  dictionary based password attack 907815 non-null  int64
 31  scanning for vulnerable software 907815 non-null  int64
 32  input data manipulation         907815 non-null  int64
 33  http verb tampering             907815 non-null  int64
 34  fake the source of data         907815 non-null  int64
 35  http response splitting         907815 non-null  int64
 36  http request smuggling          907815 non-null  int64
dtypes: float64(5), int64(13), object(19)
memory usage: 263.2+ MB
```

Fig3 . Statistical information about the dataset

Observations:

1. The dataset consists of 907,815 rows and 37 columns. It contains a mix of data types, including object (likely categorical), float64, and int64.
2. Columns like 'request_referer,' 'request_origin,' and 'request_content_type' have a significant number of null values, indicating missing data.
3. Additionally, several columns, such as 'timestamp' and 'request_body,' are likely to contain temporal and textual information, respectively.

Table 2
Number of different CAPEC classifications assigned to a web request.

Number of different CAPEC classification	Number of web requests
1	898,576
2	8132
3	1088
4	18

Before performing the training of the different machine learning models, a review and preprocessing of the dataset data is necessary to avoid inconsistent and/or duplicated data, error correction and, at the same time, to adapt the data for numerical coding so that they are usable for the machine learning models.
The main objective of the data preprocessing and selection of relevant features of the dataset is to allow the different combinations of algorithms and models evaluated to reach the maximum level of performance and efficiency in their predictions, in addition to reducing the computational cost of modeling.

Table 3
Final set of selected features and labels.

Feature	Selected
method_value	Yes
http_request_value	Yes
protocol_value	**No**
referer_value	Yes
agent_value	Yes
host_value	**No**
origin_value	**No**
cookie_value	Yes
content_type_value	**No**
accept_value	Yes
accept_language_value	**No**
accept_encoding_value	**No**
do_not_track_value	**No**
connection_value	**No**
body_value	Yes
response_http_protocol_value	**No**
http_status_code_value	**No**
http_status_message_value	Yes
response_content_length_value	Yes
000 - Normal	Yes
272 - Protocol Manipulation	Yes
242 - Code Injection	Yes
88 - OS Command Injection	Yes
126 - Path Traversal	Yes
66 - SQL Injection	Yes
16 - Dictionary-based Password Attack	Yes
310 - Scanning for Vulnerable Software	Yes
153 - Input Data Manipulation	Yes
274 - HTTP Verb Tampering	Yes
194 - Fake the Source of Data	Yes
34 - HTTP Response Splitting	Yes
33 - HTTP Request Smuggling	Yes

When working with a dataset composed of real data, it is common to have imbalanced classes; this different percentage of representation of the classes in a dataset can affect the different evaluation metrics of the learning models. Although there are several methods to generate synthetic data that promote the equal representation of the different classes in a dataset. we have chosen to work with real and evaluate the algorithms and models with specific metrics that take in consideration the different percentage of representation of the classes in the dataset: Accuracy, Precision, Recall, F-Score, Hamming Loss, Hamming Score, Jaccard Similarity and ROC AUC. The selected features and labels are detailed in Table 3.

3.3 Requirement Analysis

A good knowledge of cybersecurity and cyber attacks is a must. Further on, a hold on the ML algorithms and the general classification techniques; namely: Multilabel, Multiclass and Binary Classification goes a long way.

Cybersecurity Parameters

'request_http_method': This feature represents the HTTP method used in the request, such as GET, POST, or HEAD. In cybersecurity, abnormal or unexpected HTTP methods can indicate potential attacks. For example, detecting the use of uncommon methods or methods not typically associated with the application's functionality may suggest a malicious intent, such as attempting to exploit vulnerabilities.

'request_http_request':This likely represents the full HTTP request sent to the server. Analyzing the contents of the HTTP request can help identify unusual patterns or payloads, potentially signaling an attempted attack. Anomalies in the structure or content of the HTTP request may indicate malicious intent, such as SQL injection or cross-site scripting (XSS) attacks.

'request_cookie': This feature contains information about the cookies sent with the request. Cookies can be exploited for various attacks, such as session hijacking or Cross-Site Scripting (XSS). Monitoring and analyzing cookie values can help detect suspicious activities and protect against unauthorized access.

'request_accept': Indicates the types of content that the client can process. Anomalies in the 'request_accept' header may indicate attempts to manipulate the server's response, such as content type-based attacks. For example, an attacker might manipulate this field to receive a response in a format that could be used to exploit vulnerabilities in the application.

'response_http_status_message': This feature contains the HTTP status message returned by the server. Unusual or unexpected status messages, especially those associated with error conditions, may indicate attempts to exploit vulnerabilities or manipulate the server's behavior.

'response_content_length': This represents the length of the content in the server's response. Anomalies in content length, such as unusually large or small responses, can be indicative of various attacks, including buffer overflow attempts or data exfiltration.

These features are crucial for a cybersecurity ML model as they capture aspects of web requests and responses that are indicative of potential attacks. Analyzing these features can help the model identify patterns of behavior associated with known attack vectors and detect anomalies that may signify novel or evolving threats.

3.3.1 Functional Requirements

3.3.1.1 Input Requirements

- The code assumes the existence of a CSV file named "data_capec_multilabel.csv" containing the dataset for web attacks.
- The code uses various columns from the dataset as input features, such as 'timestamp', 'src_port', 'request_http_method', 'request_http_request', 'request_referer', 'request_user_agent', 'request_cookie', 'request_accept', 'request_body','response_http_status_message',and 'response_content_length'.

3.3.1.2 Output Requirements

- The code produces several outputs throughout its execution, including modified DataFrames, CSV files, and various visualizations.
- The final outputs include the performance metrics (accuracy, F1 score, Hamming loss, etc.) of the trained classifiers.

3.3.1.3 Processing Requirements

- The code performs data preprocessing tasks, such as renaming columns, handling missing values, and encoding categorical features.
- It uses different machine learning classifiers (Random Forest, AdaBoost, LightGBM) for multi-label classification tasks.
- The code generates visualizations using libraries like Seaborn and Matplotlib to represent the distribution of certain features and classifier performance.

3.3.2 Non-Functional Requirements

3.3.2.1 Performance Requirements

- The code uses machine learning algorithms for classification tasks, and the performance is dependent on the chosen classifiers (Random Forest, AdaBoost, LightGBM).
- The preprocessing steps, especially the handling of missing values and encoding, should efficiently process the dataset.

3.3.2.2 Security Requirement

- No specific security requirements are explicitly addressed in the code. It assumes the input data is secure and doesn't involve any security-related checks.

3.3.2.3 Usability Requirements

- The code provides visualizations to aid in understanding the distribution of features and classifier performance.
- It saves the trained classifiers (models) using joblib for future use, enhancing usability by allowing model reuse without retraining.

3.4 Implementation Tools and Analysis

Language used:

Python: 3.9.16

Implementation Tools:

Scikit-learn, Seaborn/Matplotlib, Joblib, Category_encoder, Skmultillearn, LightGBM, Catboost.

3.5 Solution Approach

The proposed solution to the identified problem is characterized by a thorough and systematic methodology, strategically crafted to effectively tackle the growing complexities associated with web attacks. This comprehensive approach is designed to ensure a robust defense mechanism against the evolving threat landscape.

A noteworthy aspect of the methodology is the application of count encoding, a technique employed to efficiently represent categorical variables. This step significantly enhances the model's ability to recognize and interpret patterns within the categorical data, contributing to a more nuanced and accurate analysis.

The machine learning models selected for this task constitute a diverse set, showcasing the adaptability and robustness of the proposed solution. The inclusion of Binary Relevance, Random Forest, Multi Output Classifier, Adaboost, LightGBM, Classifier Chain, and Labeled Power Set demonstrates a strategic fusion of models, each chosen for its unique strengths. This ensemble approach is designed to capitalize on the individual merits of each model, creating a synergistic effect that aims to provide a highly accurate and resilient defense against a broad spectrum of web attacks.

A key objective of the proposed solution is not only to identify web attacks but also to systematically classify them under the Common Attack Pattern Enumeration and Classification (CAPEC) catalog. This aligns with the overarching goal of developing a sophisticated and proactive cybersecurity framework. The integration of advanced machine learning techniques, such as feature engineering, encoding methods, and the diverse model set, contributes to the precision and efficacy of web attack detection and classification.

In summary, the solution stands out for its holistic and innovative approach. It leverages a combination of feature engineering, advanced encoding techniques, and a diverse set of machine-learning models to create a comprehensive defense mechanism. By systematically classifying identified attacks under the CAPEC catalog, the solution aligns with contemporary cybersecurity goals, aiming to not only detect but also proactively defend against the intricate tactics employed by cyber adversaries in the realm of web attacks.

Chapter - 4 Modeling and Implementation Details

Fig 4. Control Flow Diagrams

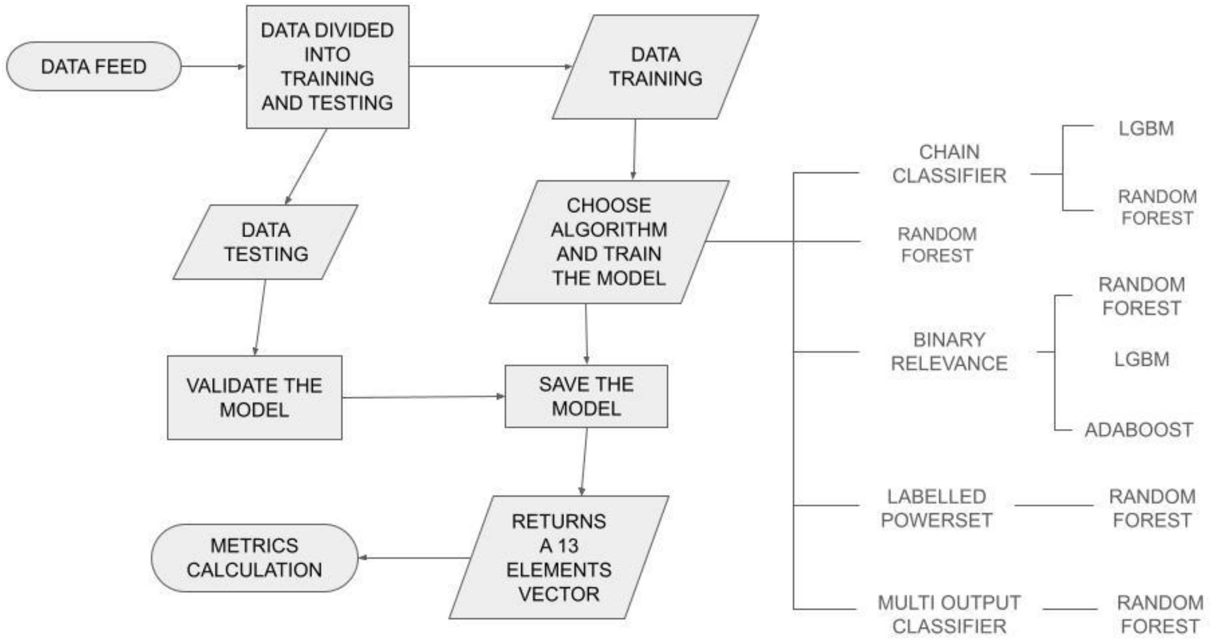
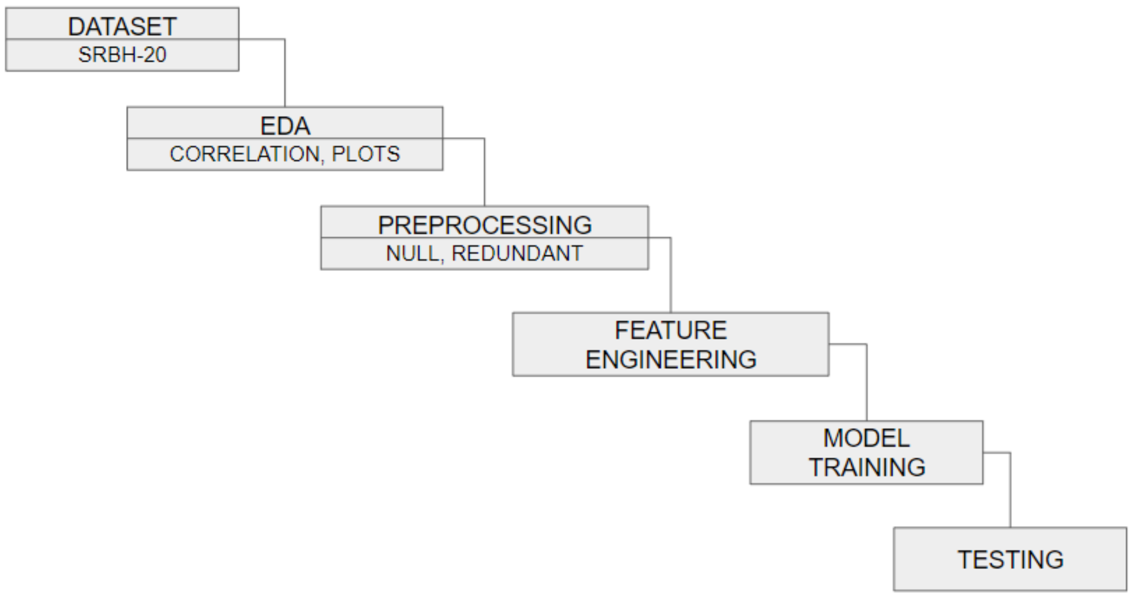


Fig 5. Sequence Diagrams

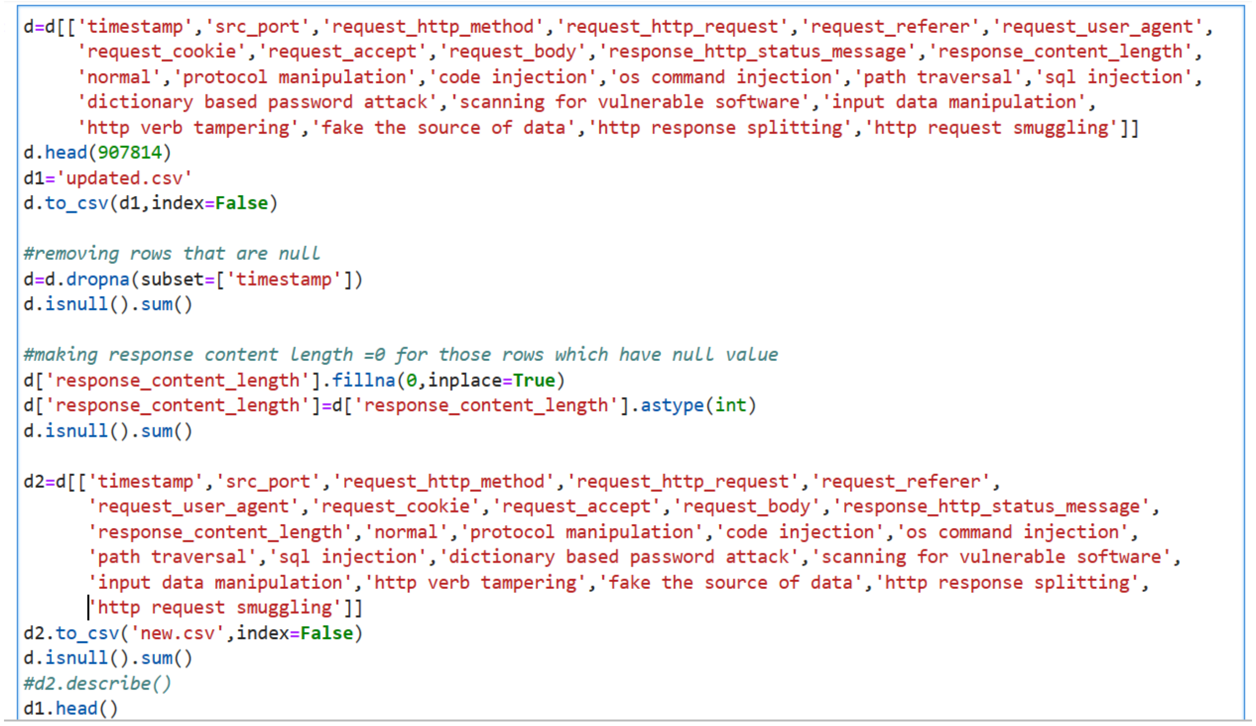


4.2 Implementation Details and Issues

Code

In fig x. We are excluding certain columns from the dataset because these columns contain information that is irrelevant to our specific problem description. Consequently, we are removing these columns as they do not contribute to solving the problem at hand.

The provided code involves data manipulation and preprocessing operations on a DataFrame, likely using the pandas library in Python. It includes column selection, displaying data, exporting to CSV files, handling null values, and converting data types. The primary focus appears to be on cleaning and organizing the data for further analysis or use. There's also a mention of a potential mistake when attempting to display the head of a DataFrame ('d1').



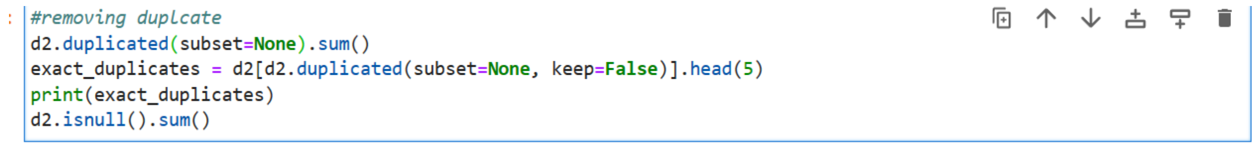
```python
d=d[['timestamp','src_port','request_http_method','request_http_request','request_referer','request_user_agent',
     'request_cookie','request_accept','request_body','response_http_status_message','response_content_length',
     'normal','protocol manipulation','code injection','os command injection','path traversal','sql injection',
     'dictionary based password attack','scanning for vulnerable software','input data manipulation',
     'http verb tampering','fake the source of data','http response splitting','http request smuggling']]
d.head(907814)
d1='updated.csv'
d.to_csv(d1,index=False)

#removing rows that are null
d=d.dropna(subset=['timestamp'])
d.isnull().sum()

#making response content length =0 for those rows which have null value
d['response_content_length'].fillna(0,inplace=True)
d['response_content_length']=d['response_content_length'].astype(int)
d.isnull().sum()

d2=d[['timestamp','src_port','request_http_method','request_http_request','request_referer',
      'request_user_agent','request_cookie','request_accept','request_body','response_http_status_message',
      'response_content_length','normal','protocol manipulation','code injection','os command injection',
      'path traversal','sql injection','dictionary based password attack','scanning for vulnerable software',
      'input data manipulation','http verb tampering','fake the source of data','http response splitting',
      'http request smuggling']]
d2.to_csv('new.csv',index=False)
d.isnull().sum()
#d2.describe()
d1.head()
```

Fig. 6 dropping irrelevant columns



```python
#removing duplcate
d2.duplicated(subset=None).sum()
exact_duplicates = d2[d2.duplicated(subset=None, keep=False)].head(5)
print(exact_duplicates)
d2.isnull().sum()
```

Fig.7 Removing Redundant Data

Fig. 2 shows how we are eliminating these columns due to the presence of exact duplicates, as they can interfere with the model training process.

Encoding

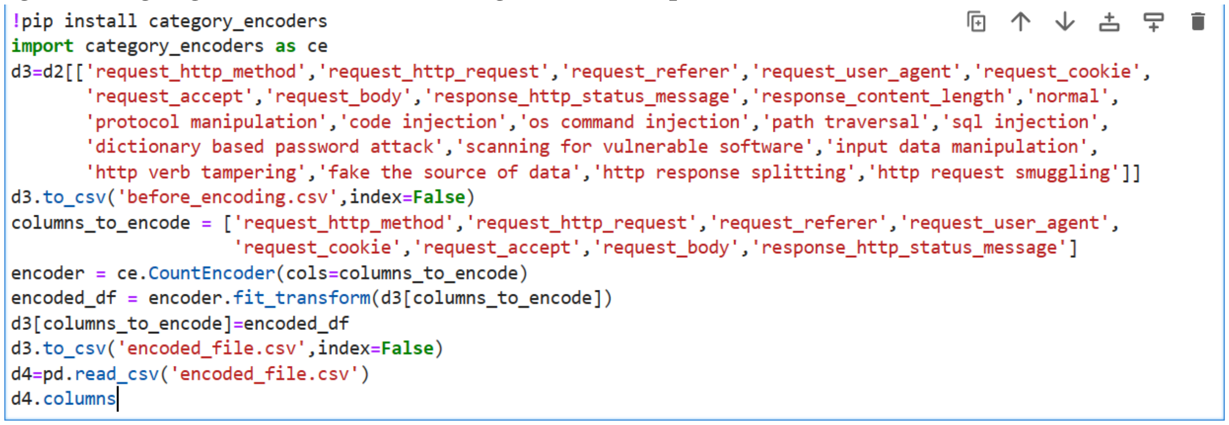
Handling High-Cardinality Features:

- Count encoding is effective when dealing with categorical features that have a high cardinality (a large number of unique categories). One-hot encoding can lead to a significant increase in dimensionality in such cases, making count encoding a more efficient alternative.

Preserving Information about Category Frequency:

- Count encoding retains information about the frequency of each category in the dataset. This can be valuable if the occurrence of categories is informative for the target variable, especially in scenarios where the frequency itself is a relevant feature.

Figure 8 Highlights how count encoding has been implemented on all the desired columns of the dataset.



```
!pip install category_encoders
import category_encoders as ce
d3=d2[['request_http_method','request_http_request','request_referer','request_user_agent','request_cookie',
       'request_accept','request_body','response_http_status_message','response_content_length','normal',
       'protocol manipulation','code injection','os command injection','path traversal','sql injection',
       'dictionary based password attack','scanning for vulnerable software','input data manipulation',
       'http verb tampering','fake the source of data','http response splitting','http request smuggling']]
d3.to_csv('before_encoding.csv',index=False)
columns_to_encode = ['request_http_method','request_http_request','request_referer','request_user_agent',
                     'request_cookie','request_accept','request_body','response_http_status_message']
encoder = ce.CountEncoder(cols=columns_to_encode)
encoded_df = encoder.fit_transform(d3[columns_to_encode])
d3[columns_to_encode]=encoded_df
d3.to_csv('encoded_file.csv',index=False)
d4=pd.read_csv('encoded_file.csv')
d4.columns
```

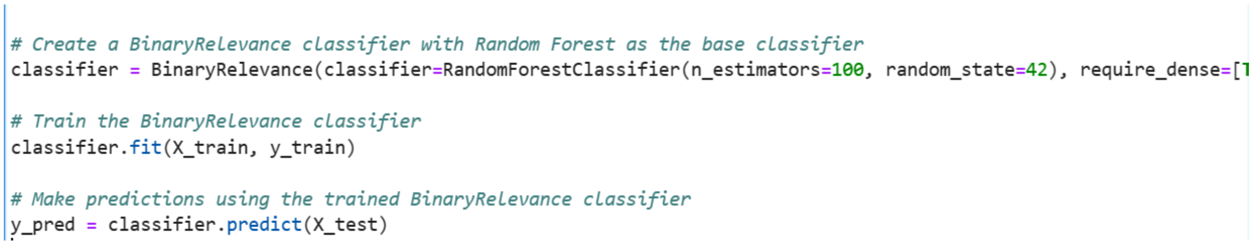
Fig. 8 Count Encoding

Models

The SR-BH 2020 dataset has a set of 13 labels. First label indicates whether the web request is considered normal or not, so it is assumed that if its value is 1 (normal request), the rest of the labels of the set should be 0. On the contrary, if the value of the first label is 0 (possible attack), there should be one or more of the labels of the remaining set with its value at 1.

This assumption allows us to establish a division of classifications by phases: Classifications can be established with a single phase model, in which an attempt is made to predict the entire
set of labels independently of the value obtained by the first label, i.e., even if the first label indicates that the request is normal, an attempt will be made to predict the remaining labels in the set. On the other hand, it is possible to generate two-phase prediction
models in which the algorithm will only predict the rest of the tags if the value of the first tag is 0; if its value is 1 (normal request), it will automatically set all the remaining tags in the set to 0.

A customized model is also generated in which the best hyperparameters for the classification of each of the labels are calculated using Binary Relevance with LightGBM and AdaBoost Algorithms.

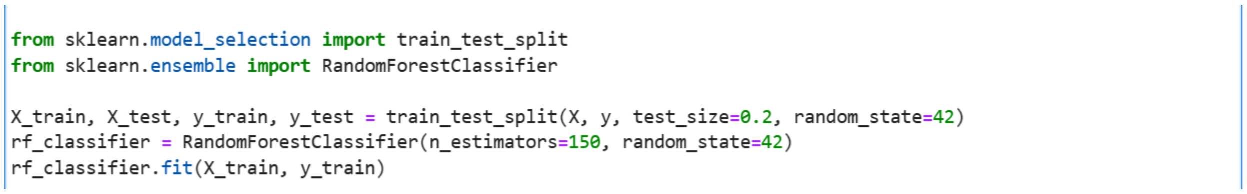


```
# Create a BinaryRelevance classifier with Random Forest as the base classifier
classifier = BinaryRelevance(classifier=RandomForestClassifier(n_estimators=100, random_state=42), require_dense=[T

# Train the BinaryRelevance classifier
classifier.fit(X_train, y_train)

# Make predictions using the trained BinaryRelevance classifier
y_pred = classifier.predict(X_test)
```

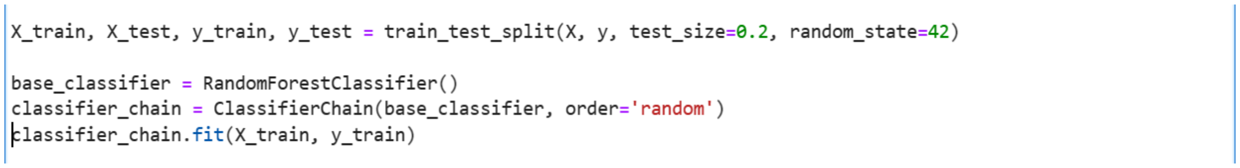
Fig: 9.1 Binary Relevance+Random Forest



```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_classifier = RandomForestClassifier(n_estimators=150, random_state=42)
rf_classifier.fit(X_train, y_train)
```

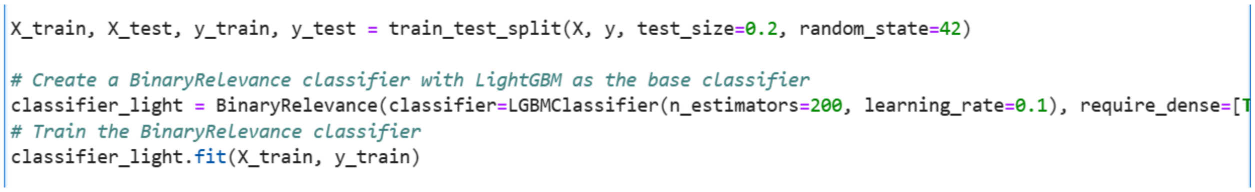
Fig: 9.2 Random Forest



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

base_classifier = RandomForestClassifier()
classifier_chain = ClassifierChain(base_classifier, order='random')
classifier_chain.fit(X_train, y_train)
```

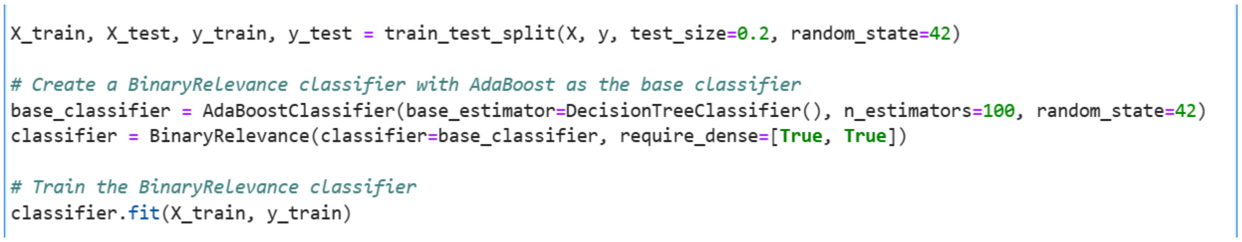
Fig 9.3 Classifier Chain+Random Forest



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a BinaryRelevance classifier with LightGBM as the base classifier
classifier_light = BinaryRelevance(classifier=LGBMClassifier(n_estimators=200, learning_rate=0.1), require_dense=[T
# Train the BinaryRelevance classifier
classifier_light.fit(X_train, y_train)
```

Fig 9.4 Binary Relevance+LGBM

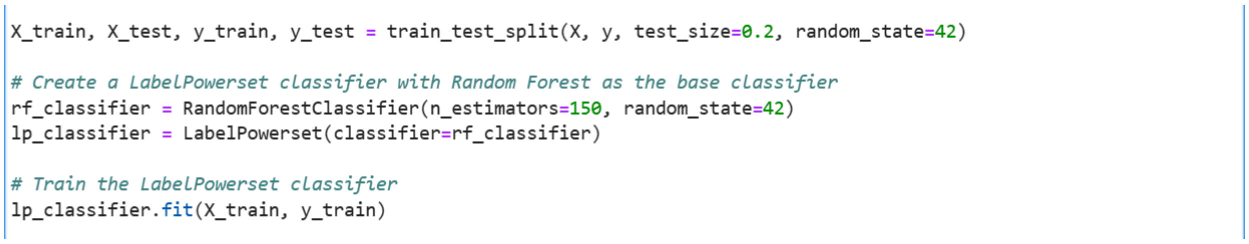


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a BinaryRelevance classifier with AdaBoost as the base classifier
base_classifier = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100, random_state=42)
classifier = BinaryRelevance(classifier=base_classifier, require_dense=[True, True])

# Train the BinaryRelevance classifier
classifier.fit(X_train, y_train)
```

Fig 9.5 Binary Relevance+AdaBoost



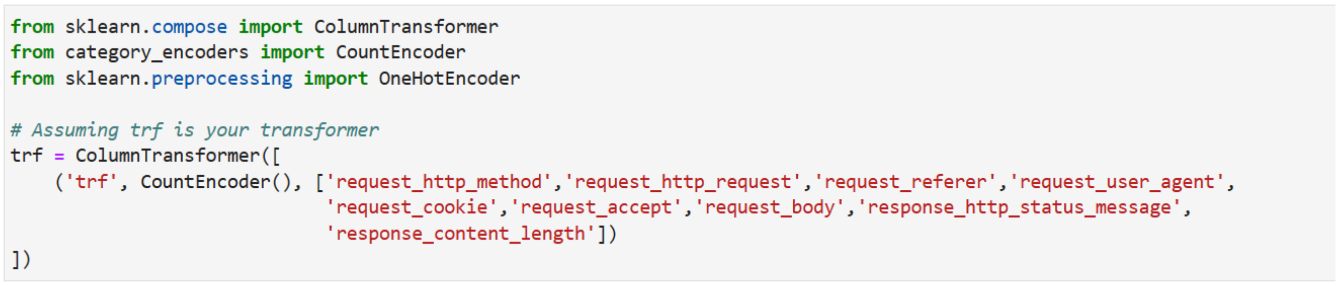
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a LabelPowerset classifier with Random Forest as the base classifier
rf_classifier = RandomForestClassifier(n_estimators=150, random_state=42)
lp_classifier = LabelPowerset(classifier=rf_classifier)

# Train the LabelPowerset classifier
lp_classifier.fit(X_train, y_train)
```

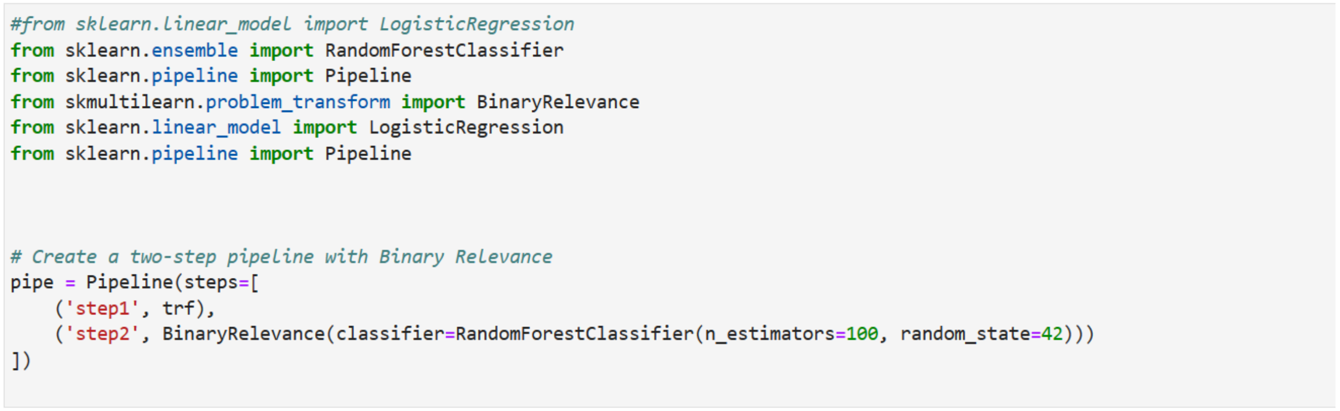
Fig 9.6 Label Powerset+Random Forest

PipeLine



```
from sklearn.compose import ColumnTransformer
from category_encoders import CountEncoder
from sklearn.preprocessing import OneHotEncoder

# Assuming trf is your transformer
trf = ColumnTransformer([
    ('trf', CountEncoder(), ['request_http_method','request_http_request','request_referer','request_user_agent',
                             'request_cookie','request_accept','request_body','response_http_status_message',
                             'response_content_length'])
])
```

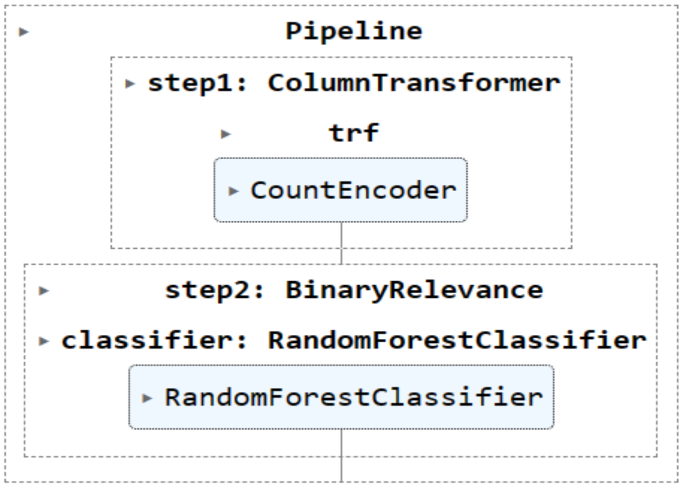


```
#from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline



# Create a two-step pipeline with Binary Relevance
pipe = Pipeline(steps=[
    ('step1', trf),
    ('step2', BinaryRelevance(classifier=RandomForestClassifier(n_estimators=100, random_state=42)))
])
```

Fig 10. Creating a pipeline



Algorithm
Here we have used multiple machine learning algorithms to test and predict whether a given URL is a cyber attack or not, and if so, under which classification it belongs to.

1. Binary Relevance: Binary Relevance is a simple and straightforward method for multi-label classification. In this approach, each label is treated as a separate binary classification problem. For each label, a binary classifier (e.g., logistic regression, support vector machine) is trained independently. The final prediction is then obtained by combining the predictions of all individual classifiers.

2. Random Forest: Random Forest is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It's effective for handling categorical data because decision trees inherently handle categorical variables, and the ensemble nature of Random Forest helps reduce overfitting and improves generalization.

3. Adaboost: AdaBoost (Adaptive Boosting) is an ensemble learning method that focuses on improving the performance of weak learners (e.g., shallow decision trees). It works by assigning weights to data points and adjusting them at each iteration to emphasize the misclassified points. The weak learners are then combined to form a strong learner. AdaBoost is versatile and can be used with various base classifiers, making it suitable for handling categorical data.

4. LightGBM: LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It's designed for distributed and efficient training of large datasets. LightGBM is particularly efficient for categorical features, as it supports categorical data natively without the need for one-hot encoding. It uses a histogram-based learning approach that speeds up training and reduces memory usage.

5. Classifier Chain: Classifier Chain is a method for multi-label classification where a chain of binary classifiers is created, and each classifier is trained to predict one label as well as the labels that precede it in the chain. This method can capture label dependencies, as the prediction for one label is influenced by the predictions of previous labels in the chain.

6. Labeled Power Set: Labeled Power Set is an approach to multi-label classification where each unique combination of labels is treated as a separate class. This method transforms the multi-label problem into a multi-class problem. It requires a significant number of classes, which may lead to a high-dimensional problem.

7. Multi-Output Classifier: A Multi-Output Classifier is a type of model that can handle multiple output variables, each of which may be associated with different labels. It's a versatile approach that can be applied to various types of models, including decision trees, neural networks, and others. It can naturally handle categorical data in its input features and output labels.

Combining These Algorithms: To build a machine learning model on categorical data, you can use a combination of these algorithms in different ways. For example, you might preprocess categorical features using techniques like one-hot encoding or label encoding before applying Binary Relevance, Random Forest, Adaboost, or LightGBM.

4.3 Risk Analysis and Mitigation

Data Quality and Availability:
- Risk: Inadequate or poor-quality data can negatively impact model performance.
- Mitigation Strategies: Conduct thorough data quality assessments, implement data cleaning and preprocessing techniques, and have backup plans for handling missing or incomplete data. Consider data augmentation methods if necessary.

Computational Resources:
- Risk: Insufficient computational resources may lead to long training times or prevent the use of complex models.
- Mitigation Strategies: Estimate and plan for the necessary computational resources. Consider distributed computing or cloud services to scale resources as needed.

Security Concerns:
- Risk: ML models can be vulnerable to attacks, and sensitive data may be at risk.
- Mitigation Strategies: Implement security best practices, encrypt sensitive data, and regularly update software dependencies. Conduct security audits and penetration testing.

Chapter - 5 Testing

5.1 Testing Plan

A testing plan is a document that outlines the strategy, scope, resources, schedule, and activities for testing a specific system or application. In the context of your web attacks classification project, a testing plan could include the following components:

1. Objective:
Clearly state the purpose of the testing plan, such as validating the performance and accuracy of the web attacks classification model.

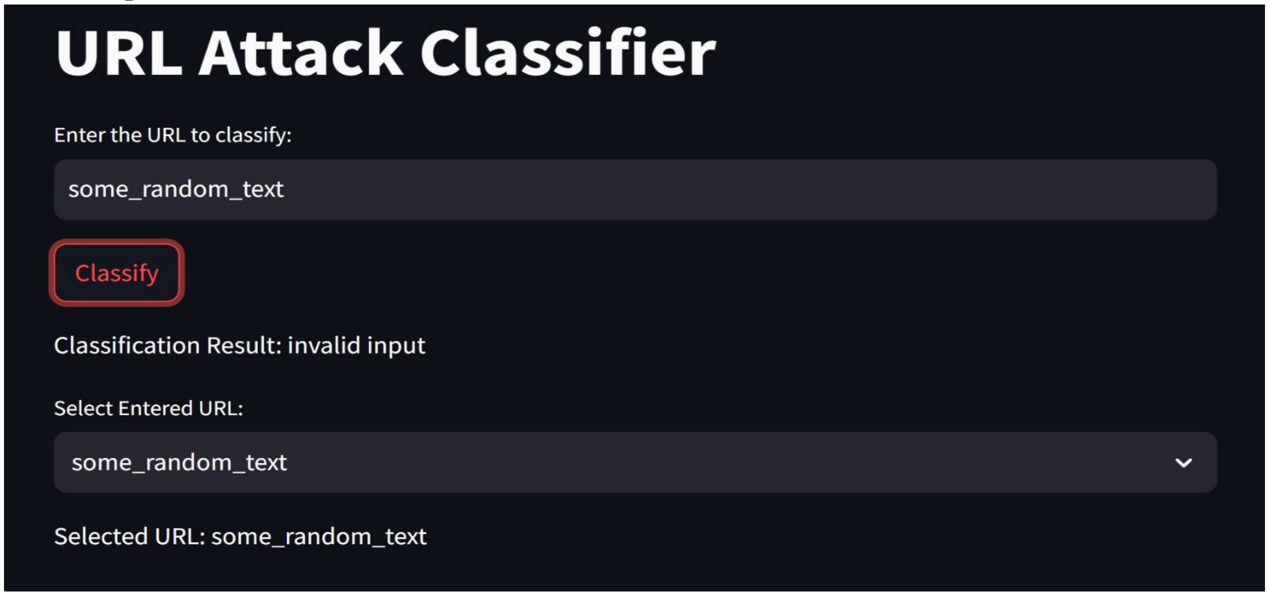
2. Scope:
Define the scope of the testing, specifying which aspects of the system will be tested. In your case, it could include evaluating the model's performance on a test dataset, assessing its ability to correctly classify various types of web attacks.

3. Testing Types:
Identify the types of testing that will be performed. This may include:
    - Unit Testing: Testing individual components or functions of the code.
    - Integration Testing: Verifying the interaction between different components.
    - System Testing: Evaluating the entire system's functionality.

5.2 Test cases

Following figures highlights the testing of the model and UI under various cases:

Invalid Input


URL Attack Classifier

Enter the URL to classify:

some_random_text

Classification Result: invalid input

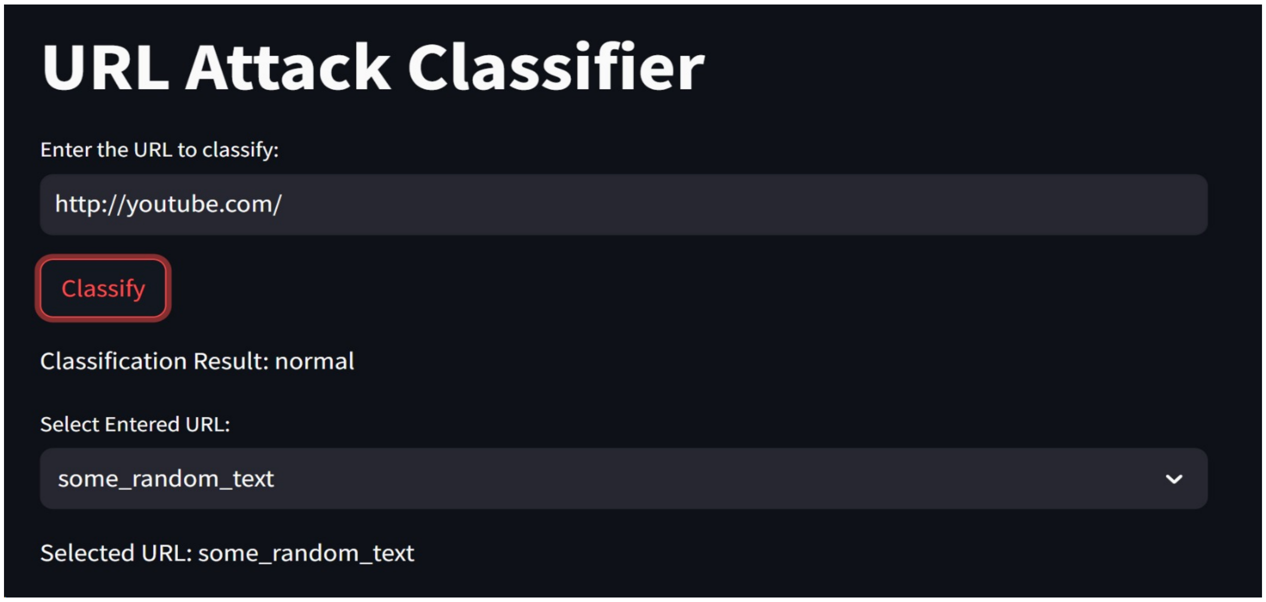
Select Entered URL:

some_random_text

Selected URL: some_random_text

Fig 11.1 Testing user inputs - Showing Invalid Input

Normal



URL Attack Classifier

Enter the URL to classify:

http://youtube.com/

Classification Result: normal

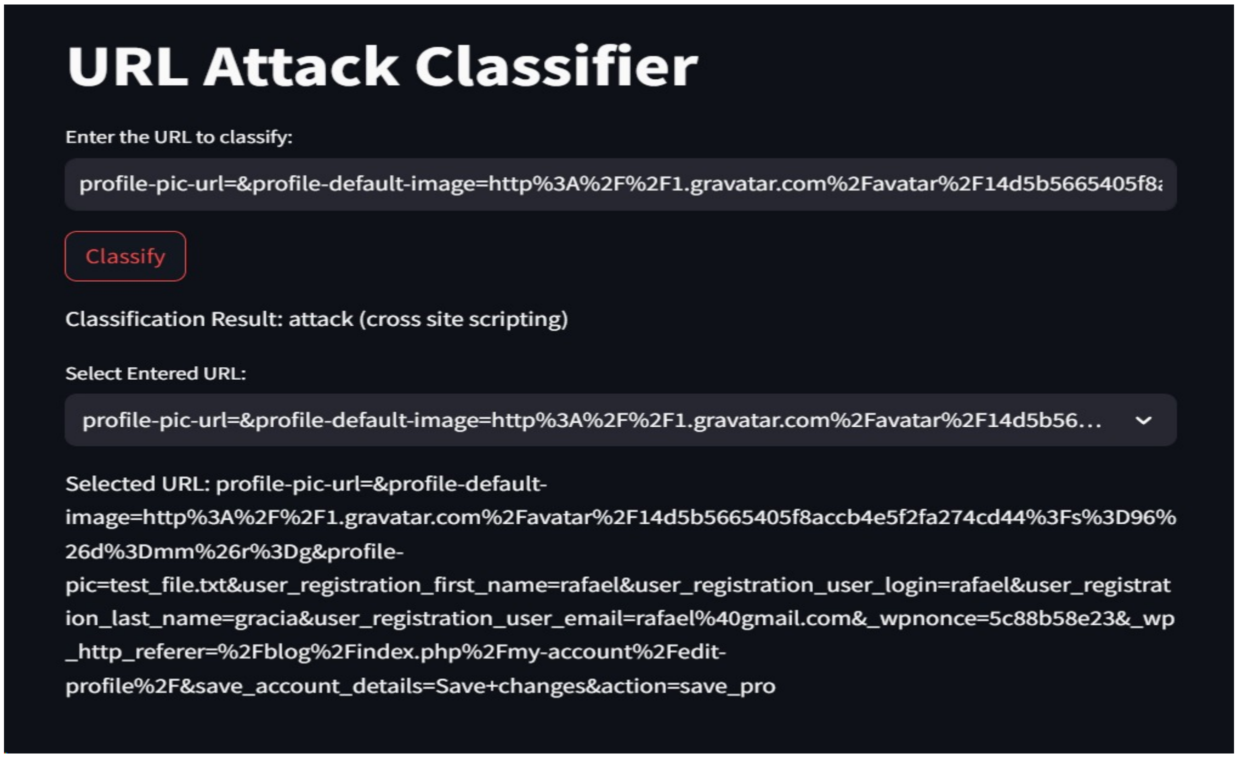
Select Entered URL:

some_random_text

Selected URL: some_random_text

Fig 11.2 Testing user inputs - Normal URL(Not an attack)

Attack



URL Attack Classifier

Enter the URL to classify:

profile-pic-url=&profile-default-image=http%3A%2F%2F1.gravatar.com%2Favatar%2F14d5b5665405f8a

Classification Result: attack (cross site scripting)

Select Entered URL:

profile-pic-url=&profile-default-image=http%3A%2F%2F1.gravatar.com%2Favatar%2F14d5b56...

Selected URL: profile-pic-url=&profile-default-image=http%3A%2F%2F1.gravatar.com%2Favatar%2F14d5b5665405f8accb4e5f2fa274cd44%3Fs%3D96%26d%3Dmm%26r%3Dg&profile-pic=test_file.txt&user_registration_first_name=rafael&user_registration_user_login=rafael&user_registration_last_name=gracia&user_registration_user_email=rafael%40gmail.com&_wpnonce=5c88b58e23&_wp_http_referer=%2Fblog%2Findex.php%2Fmy-account%2Fedit-profile%2F&save_account_details=Save+changes&action=save_pro

Fig 11.2 Testing user inputs - Attack URL

5.3 Limitations of the solution

Sensitivity to Rare Categories:

Count encoding is sensitive to rare categories. If a category occurs infrequently, its count will be low, possibly leading to overfitting. Rare categories may not have enough information to be reliably generalized.

Increased Dimensionality:

Count encoding can increase the dimensionality of the dataset, especially when dealing with categorical features with a large number of unique values. This may impact the efficiency of some machine learning algorithms.

Independence Assumption:

Binary Relevance assumes that the labels are independent of each other. In real-world scenarios, this assumption may not always hold, and the relationships between labels might affect the model's performance.

Imbalance in Class Distribution:

Random Forest struggles when dealing with imbalanced class distributions in multi-label problems.Some labels are significantly more frequent than others, thereby increasing the possibility that algorithm may be biased towards the majority class, leading to suboptimal performance for minority classes.

Chapter - 6 Findings, Conclusions, and Future Work

6.1 Findings

In the assessment of machine learning models, the selection of appropriate evaluation metrics is crucial to comprehensively understand their performance. The chosen parameters for analyzing the results of this minor project are Accuracy, F1 Score (Micro), Hamming Loss, F1 Score (Macro), Precision (Micro), Precision (Macro), Recall (Micro), Recall (Macro), Micro-Averaged ROC AUC, and Micro-Averaged Jaccard Similarity. Each of these metrics serves a distinct purpose in evaluating different aspects of the model's performance.

1. Accuracy

   Accuracy is a common evaluation metric for classification models. It represents the ratio of correctly predicted instances to the total instances in the dataset.

   Calculation:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

   Accuracy provides a fundamental measure of the model's overall correctness in predicting all classes.

2. F1 Score (Micro):

   The F1 Score is a metric that combines precision and recall. The micro-averaged F1 score is particularly useful in multi-label classification scenarios.

   Calculation:

$$\text{F1 Score (Micro)} = \frac{2 \times \text{Precision (Micro)} \times \text{Recall (Micro)}}{\text{Precision (Micro)} + \text{Recall (Micro)}}$$

   Micro-averaged F1 Score balances precision and recall, especially useful in multi-label classification scenarios, and is well-suited for imbalanced datasets.

3. Hamming Loss:

   Hamming Loss measures the fraction of labels that are incorrectly predicted. It is particularly relevant in multi-label classification where each instance can belong to multiple classes.

   Calculation:

$$\text{Hamming Loss} = \frac{1}{N} \sum_{i=1}^{N} \frac{(\text{Number of Incorrectly Predicted Labels for instance } i)}{(\text{Total Number of Labels for instance } i)}$$

   Hamming Loss is pertinent in multi-label classification, quantifying the fraction of incorrectly predicted labels, and offering insights into the model's label-wise accuracy.

4. Precision (Micro):

Micro-averaged precision is a metric that considers all instances and calculates precision across all classes collectively. Micro-averaged precision considers all instances collectively, providing an indication of the model's ability to avoid false positives across all classes.

Calculation:

$$\text{Precision (Micro)} = \frac{\sum_{k-1}^{K} \text{True Positives for Class } k}{\sum_{k-1}^{K} (\text{True Positives for Class } k + \text{False Positives for Class } k)}$$

5. Recall (Micro):

Micro-averaged recall considers all instances and calculates recall across all classes collectively. Micro-averaged recall considers all instances collectively, providing insights into the model's ability to capture all true positives across all classes.

Calculation: $$Recall = \frac{TP}{TP + FN}$$

$$Recall_{weighted} = \sum_{classes} weight\ of\ class\ x\ recall\ of\ class$$

6. ROC AUC:

Shows the global efficiency of a classification model at all classification levels, by plotting the true positive rate (TPR) versus the false positive rate (FPR). In our case, ROC AUC is averaged across all instances and support-weighted. ROC AUC is particularly valuable for binary and multilabel classification, providing an aggregated measure of the model's ability to discriminate between positive and negative instances.

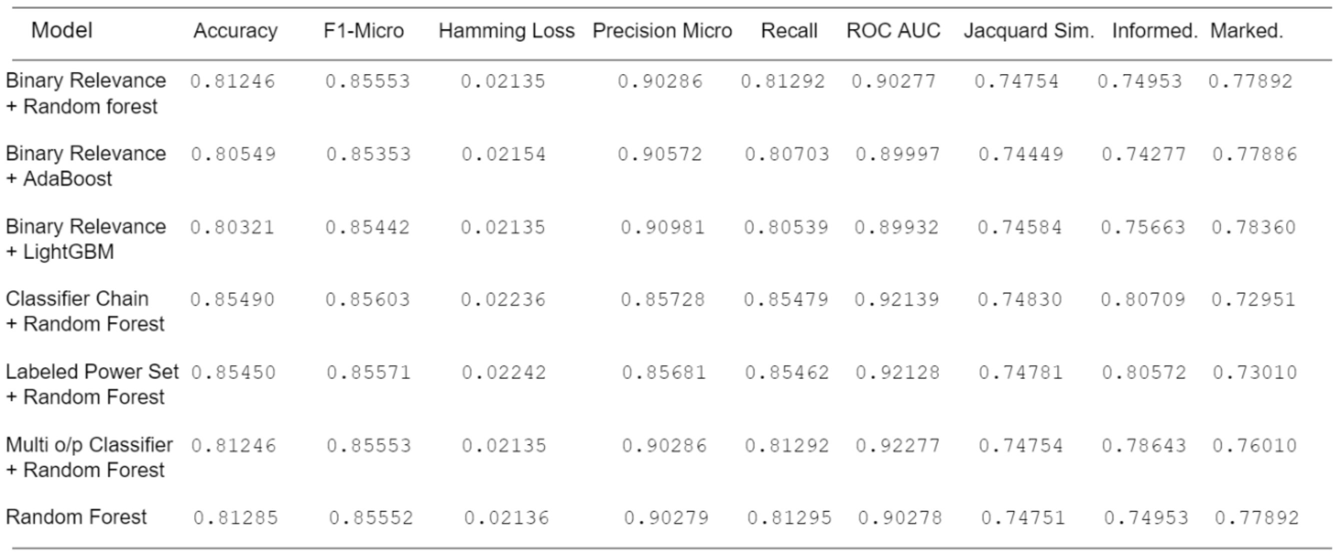
7. (Micro-Averaged) Jaccard Similarity:

It measures the degree of similarity between two sets by examining the proportion of correctly predicted positive labels in a potentially positive set (expected positive and real positive).

Calculation: $$\mathcal{J}(T, P) = \frac{T \cap P}{T \cup P}$$

The chosen metrics collectively offer a comprehensive view of the model's performance, considering aspects such as overall correctness, label-wise accuracy, and the balance between precision and recall. This selection ensures a thorough evaluation, essential for understanding the strengths and limitations of the machine learning model developed in this minor project. Table 6. Gives a comprehensive summary of the metrics, algorithm wise.

Table 4
Summary of metrics by algorithm and model, ordered by recall score.



Model	Accuracy	F1-Micro	Hamming Loss	Precision Micro	Recall	ROC AUC	Jacquard Sim.	Informed.	Marked.
Binary Relevance + Random forest	0.81246	0.85553	0.02135	0.90286	0.81292	0.90277	0.74754	0.74953	0.77892
Binary Relevance + AdaBoost	0.80549	0.85353	0.02154	0.90572	0.80703	0.89997	0.74449	0.74277	0.77886
Binary Relevance + LightGBM	0.80321	0.85442	0.02135	0.90981	0.80539	0.89932	0.74584	0.75663	0.78360
Classifier Chain + Random Forest	0.85490	0.85603	0.02236	0.85728	0.85479	0.92139	0.74830	0.80709	0.72951
Labeled Power Set + Random Forest	0.85450	0.85571	0.02242	0.85681	0.85462	0.92128	0.74781	0.80572	0.73010
Multi o/p Classifier + Random Forest	0.81246	0.85553	0.02135	0.90286	0.81292	0.92277	0.74754	0.78643	0.76010
Random Forest	0.81285	0.85552	0.02136	0.90279	0.81295	0.90278	0.74751	0.74953	0.77892

The results of the algorithms and model combinations have been evaluated according to the metrics and scenarios discussed in Section 6.1. One of the main objectives of the present work is to propose the best combination of algorithms and models able to classify as accurately as possible the different types of attacks in each possible scenario, following the CAPEC classification. Since a web request can be simultaneously classified in more than one CAPEC key, it is necessary to work with multi-label models.

6.2 Conclusions

In this work we have presented the SR-BH 2020 multi-label dataset, which includes a set of 13 different labels, providing information about the normality of each web request and its possible classification into 12 different CAPEC categories. A new way to give a numerical value to the alphanumeric strings and symbols that constitute the different fields that conform a web request, by calculating the average of the sum of the ASCII values of each of the characters in each field, is proposed; this numerical value, calculated easily and quickly, allows the extraction of features and the training of the different machine learning models. We have also designed and evaluated different multi-label classification models, using modules and classes from the sci-kit learn and scikit-multi learn libraries. Two leading algorithms in the field of machine learning have been tested with these models: LightGBM and AdaBoost. The results obtained by our experiments show a clear superiority of the combination of the CatBoost algorithm and the two-phase model with the MultiOutputClassifier module of the scikit-learn library, in multi-label classification tasks. In future work, the possibility of executing automatic remediation actions, based on CAPEC attack patterns, could be considered.

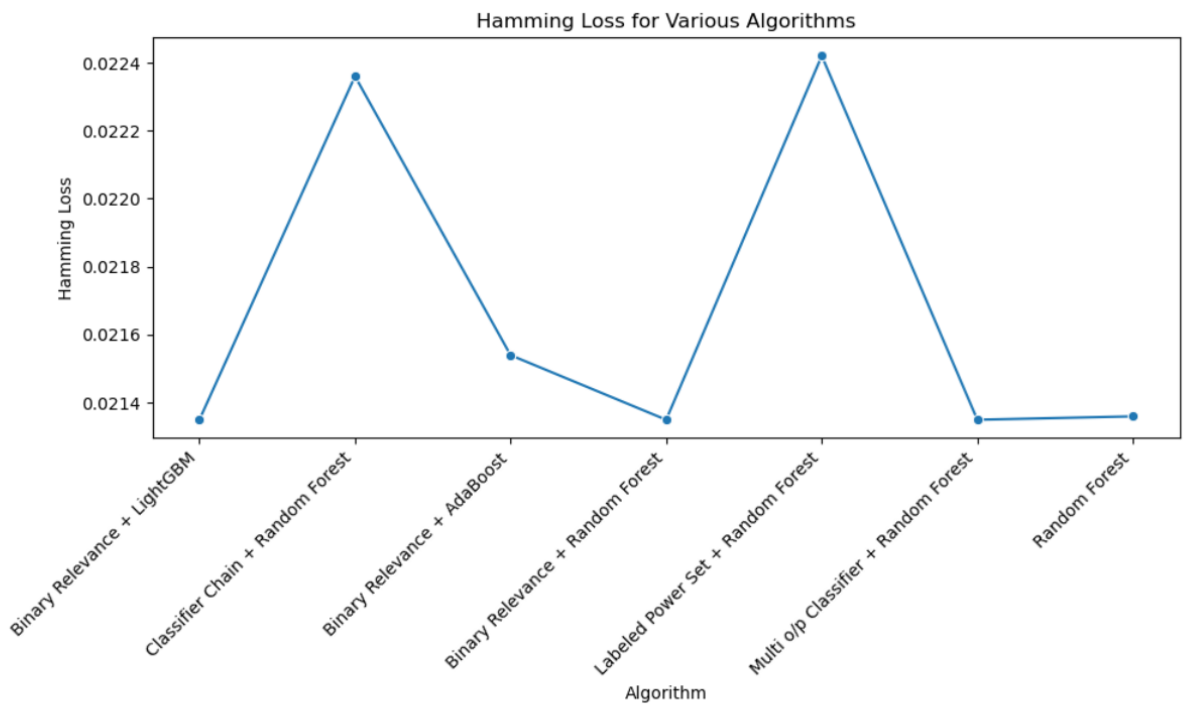


Fig 12 Hamming Loss by algorithms

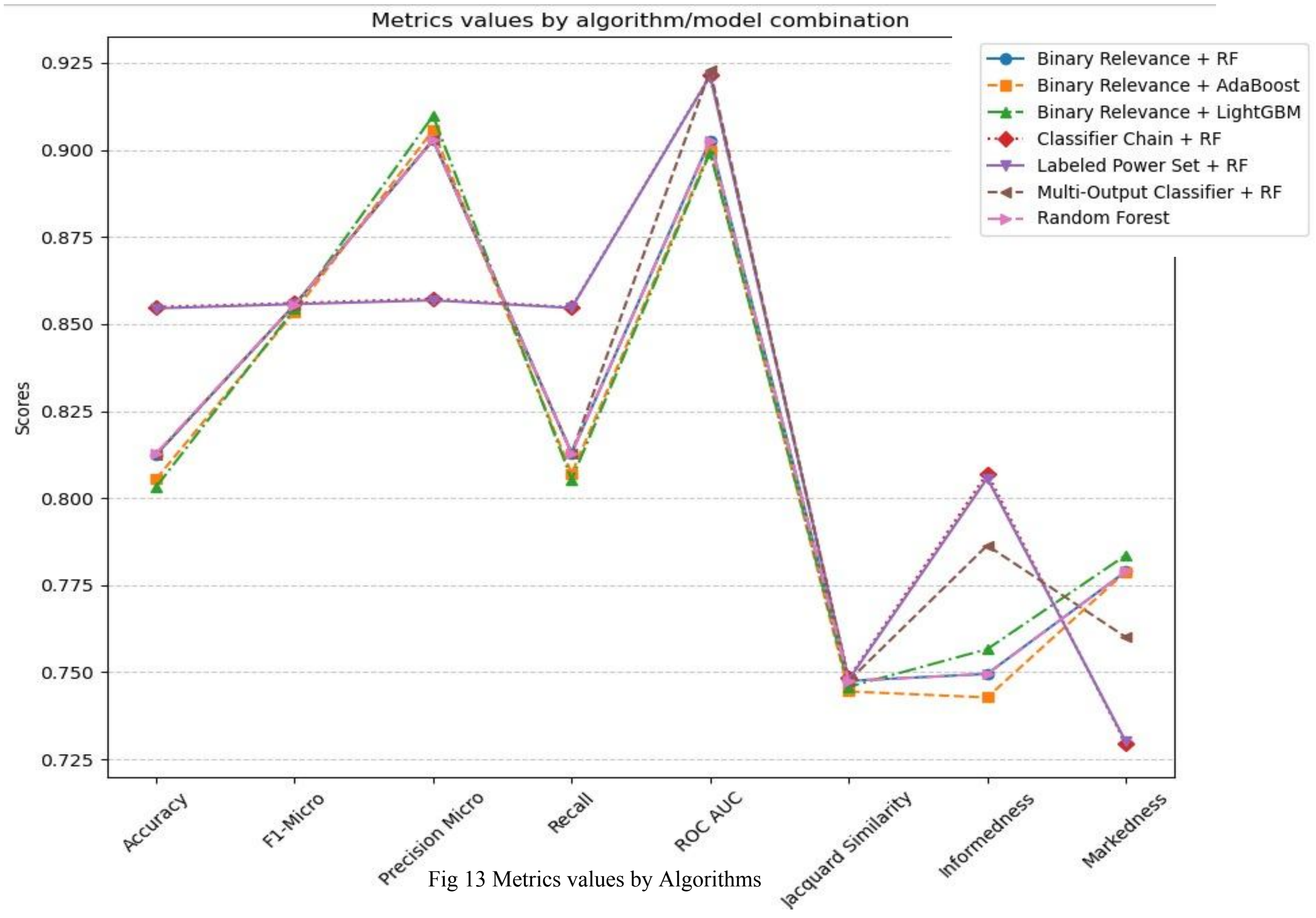


Fig 13 Metrics values by Algorithms

6.3 Future Work

Cross-Validation Strategies:

- Experiment with different cross-validation strategies to ensure robustness of your model evaluations. Stratified sampling or time-based splitting, depending on the nature of your data, could be explored to obtain more reliable performance estimates.

Hyperparameter Tuning:

- Conduct a more extensive hyperparameter tuning process for the selected algorithms. This can involve using techniques like grid search, random search, or more advanced optimization algorithms to fine-tune model parameters and improve overall performance.

Deep Learning Approaches:

- Evaluate the performance of deep learning models, such as neural networks or deep neural networks, on your dataset. Deep learning architectures may uncover intricate patterns in the data that traditional machine learning algorithms might miss.

Model Deployment and Monitoring:

- If applicable, explore the challenges and considerations for deploying your model in a production environment. Consider implementing monitoring mechanisms to track model performance over time and ensure its continued effectiveness.

External Datasets and Transfer Learning:

- Explore the possibility of incorporating external datasets that might provide additional relevant information for your classification task. Transfer learning techniques could be considered if pre-trained models on related tasks are available.

We will also consider ‘limitations of the solution’ points discussed before.

References

[1] Antunes, N., Vieira, M., 2015. On the metrics for benchmarking vulnerability detection tools. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 505–516. doi:10.1109/DSN.2015.30.

[2] Auxilia, M., Tamilselvan, D., 2010. Anomaly detection using negative security model in Web application. In: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 481–486. doi:10. 1109/CISIM.2010.5643461.

[3] Bermejo Higuera, J.R., 2013. Metodología de evaluación de herramientas de análisis automático de seguridad de aplicaciones web para su adaptación en el ciclo de vida de desarrollo. Universidad Nacional Educación a Distancia (UNED). http://e-spacio.uned.es/fez/eserv/tesisuned:IngInd-Jrbermejo/ BERMEJO_HIGUERA_Juan_Ramon_Tesis.pdf.

[4] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and regression trees.(the wadsworth statistics/probability series), belmont. CA: Wadsworth.

[5] Brugger T.. KDD Cup '99 dataset (Network Intrusion) considered harmful (KDnuggets News 07:18, item 4, Features). 2007. https://www.kdnuggets.com/news/2007/ n18/4i.html.

[6] Riera, T. S., Higuera, J. R. B., Higuera, J. B., Herraiz, J. J. M., & Montalvo, J. A. S. (2022). A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques. Computers & Security, 120, 102788.

[7] Bogatinovski, J., Todorovski, L., Džeroski, S., & Kocev, D. (2022). Comprehensive comparative study of multi-label classification methods. Expert Systems with Applications, 203, 117215.