BSCS-S25-002

03-134221-037 SYED MUHAMMAD IBRAHIM

03-134221-002 ABDULLAH TAHIR

ThreatLens: A Cutting-Edge AI Model for Analyzing Hidden Network Trends & Patterns

In partial fulfilment of the requirements for the degree of
Bachelor of Science in Computer Science

Supervisor: Dawood Akram

Department of Computer Sciences

Bahria University, Lahore Campus

January 2026

C e r t i f i c a t e

We accept the work contained in the report titled

"ThreatLens: A Cutting-Edge AI Model for Analyzing Hidden Network Trends & Patterns"

written by

SYED MUHAMMAD IBRAHIM

ABDULLAH TAHIR

as a confirmation to the required standard for the partial fulfilment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: DAWOOD AKRAM

_____ (Signature)

January 05, 2026

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-134221-037	SYED MUHAMMAD IBRAHIM	
03-134221-002	ABDULLAH TAHIR	

Date : January 05, 2026

Specially dedicated to

my beloved Grandparents, Parents, Siblings, Family and Friends

Syed Muhammad Ibrahim

ACKNOWLEDGEMENTS

We would like to thank everyone who had contributed to the successful completion of this project. We would like to express our gratitude to our research supervisor, *Mr. Dawood Akram* and *Mr. Abdul Manan* CEO of *Genesis Technologies* for their invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, we would also like to express my gratitude to our loving grandparents, parents, family and friends who had helped and given us encouragement.

SYED MUHAMMAD IBRAHIM
ABDULLAH TAHIR

ThreatLens: A Cutting-Edge AI Model for Analyzing Hidden Network Trends & Patterns

ABSTRACT

The world is witnessing a growing problem of cybersecurity as the number of threat actors that use backdoor networks and other communication tunnels to strategize, organize, and conduct advanced cyberattacks continues to increase. The security solutions are mostly reactive as they react to the occurrence of incidents when they have already occurred and caused damage to critical infrastructure.

This project proposes a novel automated threat intelligence system that is specifically designed to close this fundamental security gap by availing proactive predictions of any upcoming threats to any specific industry. This contrasts with other traditional security tools, which react to threats only after they occur and become actual attacks.

This system has a technical architecture that integrates advanced web crawling technology, Data Preprocessing pipeline specially tailored for this project, and artificial intelligence measures to have a threat intelligence technology fully automated. The solution uses custom-written crawling scripts that automatically search hidden network infrastructure with proper anonymization and access methods and systematically gather data with help of data processing pipeline that is associated to threat-related information on the hidden network sites which post data leaks. This curated data is then trained into a machine learning model that allows the system to automatically classify which specific industry can get affected in the upcoming time. The AI model uses natural language processing to gain context and extract useful information from the sites and is coupled with pattern recognition algorithms to determine specific information like URLs, dates, data sizes etc. And give proactive alert using dedicated Telegram channel.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CatBoost	A machine learning gradient boosting algorithm
CTI	Cyber Threat Intelligence
CSV	Comma-Separated Values
DOM	Document Object Model
EDA	Exploratory Data Analysis
Flask	Web Application Framework used for frontend integration
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IOCs	Indicators of Compromise
IoT	Internet of Things
JSON	JavaScript Object Notation
ML	Machine Learning
NER	Name Entity Recognition
NLTK	Natural Language Processing Toolkit
NLP	Natural Language Processing
OSINT	Open-Source Intelligence
Regex	Regular Expression
SDLC	System Development Lifecycle
SMOTE	Synthetic Minority Over-Sampling
TF-IDF	Term Frequency-Inverse Document Frequency
TOR	The Onion Router
UI	User Interface
URL	Uniform Resource Locator

CHAPTER 1

INTRODUCTION

1.1 Background

Genesis Technologies is a leading solution provider of threat intelligence and security operations platforms. ***Orion Intelligence*** is the flagship product of the company which is a security intelligence platform and currently providing services to various national and international organizations.

With the passage of time *Genesis Technologies* found a critical gap in capabilities of their platform, the absence of proactive threat prediction. As a part of this project, we proposed them if we can make them this module which will be classifying hidden network sites, crawl them, get data from them and train an AI model which will proactively predict and inform the users of *Orion Intelligence* about any upcoming threats to a specific industry. Due to lack in this proactiveness Orion was unable to inform their users about emerging threats, breaches and vulnerabilities.

In response to the above mentioned this project was initiated as an industrial collaboration in which we will be providing an AI model which will be integrated with the Orion Intelligence to fill the gap of the proactive alerts.

1.2 Problem Statements

The fundamental problem Orion Intelligence facing is lack of proactiveness, clients wanted a solution which will let them know in advance about any upcoming threats and attacks as we are currently providing reactive rather than proactive solution. Breach detection gets detected in approximately 280 days [1] and, in the meantime,

attackers share the leaked data on the hidden networks they wanted a solution to proactively alert organizations to protect and update their security measures.

The main issue in this solution was that there is no dataset available for this kind of problem but as we had access to a lot of hidden networks sites, we managed to crawl them using Playwright, REQUESTS and other crawling and preprocessing methods and extracted the information we needed to train an AI model for the solution of this problem. Also, this will send an intimation message to our dedicated Telegram channel through which our users can stay updated regarding the latest threats and breaches being published on the hidden networks.

1.3 Aims and Objectives

The main goals of this project include:

- To classify hidden network or Clearnet sites to know if these sites are useful or not for us in the threat prediction.
- To form datasets from leaks related sites for model training of threat prediction.
- To train an AI model from the datasets we have created that will predict any upcoming threats to organizations or industries (Automobile, Technology, etc)
- To create an Autonomous Telegram channel which will notify about any sort of upcoming threats to industries (Automobile, Technology, etc)

1.4 Scope of Project

ThreatLens's scope is to train an AI Model for analyzing hidden network trends and Patterns. The system will predict what kind of companies or industry have chances to face data breaches. The system implements specialized web crawling capabilities designed to navigate hidden network sites anonymously. ThreatLens systematically crawls hidden network sites related to leaks and get the data from it, data may include name of company being compromised, date of breach, description of company, details of breach, downloadable compromised data, amount of data being compromised and many more. The collected data is then passed through a complete preprocessing pipeline (cleaning, scaling, etc) which is mandatory for creating a good and meaningful dataset. Also, we have applied Rule Based NLP for extracting specific information from the scraped data.

CHAPTER 2

LITERATURE REVIEW

2.1 ThreatLens: Overview

ThreatLens is a core module for Orion Intelligence which will add the up to its power all new feature of proactive predictions of upcoming threats, alongside with the capability of Website classification which was a core issue for the OSINT team in the company, due to the vast diversity among the hidden network sites [2].

We have created our very own dataset for both website classification and threat prediction models as we needed it to be compatible with the data and sites present on the hidden network (Dark Web); all the dataset requirements were thoroughly discussed with the company.

2.1.1 Existing Solution

There are solutions, but they are not compatible with for this kind of problem which could have been integrated with the Orion Intelligence as most of the research work is on the data of surface web [3], which is not suitable for projects like this, however we have made this possible by starting the process from scratch and tailored all the approaches as per hidden network and the requirement of our main project.

2.1.2 Limitation of Existing solutions

The rapid digital transformation of modern society has led to an unprecedented increase in the complexity and frequency of cyber threats. As organizations become more interconnected with the help of cloud infrastructures, Internet of Things (IoT) devices, and remote work systems, cybercriminals are leveraging automation and artificial intelligence (AI) to develop highly sophisticated and adaptive attacks. These advanced cyber threats ranging from polymorphic malware to zero-day exploits pose severe challenges to traditional cybersecurity mechanisms that rely primarily on static rules and reactive security system.[4].

In this all if we have still reactive solutions, that's not enough that was the core reason for Orion Intelligence to take this step and came up with this proactive solution.

2.2 Related Work

Mostly crawlers are based on the sites indexed on surface web, so what about the data on the hidden networks? Building a hidden network crawler is not something that no one has tried, in ***Standford University***, Computer Science department a hidden network Website classifier was built named as *HiWE* (**Hi**dden **W**eb **E**xposer) which was a general scrapper and they found the coverage of the entire hidden network very difficult [5].

There is an autonomous hidden network crawlers just like *Orion Intelligence;* this method of proactive threat hunting by overlaying a centralized threat database with enriched indicators of compromise (IOCs) allows responding to incidents and take down domains faster, concluding that this paradigm shift in defence would greatly improve cyber security preparedness [6].

Natural Language Processing (NLP) is a vital application of threat prediction because it helps organizations to pre-empt and react to cyber threats in advance by

using Cyber Threat Intelligence (CTI). NLP in CTI is applied to the unstructured textual data of a large volume, including security reports, social media, and dark web boards, through an automated procedure to detect the patterns of threat and support the collection of intelligence in real-time [7].

In certain instances, rule-based NLP was employed during the development of ThreatLens because it was mandatory to the various type of data that existed on the hidden network since most of the AI Models were providing so poor results on the hidden network. Rule-Based NLP was used as a game changer. A conventional method of Named Entity Recognition (NER) in cybersecurity is rule based methods. These pioneer NER systems are based on rules and syntactic lexical patterns that are created by experts to extract entities in text [8] [9] .

Multiple AI Models usage in scenarios like hidden network related tasks is considered as a good practice as it helps the organization or the user to cross check from various models, in our scenario we have used 5 AI Models for Website classification purposes for comparative analysis and results. Models include Naïve Bayes, XGBoost, CatBoost, Random Forest Classifier, Gradient Boosting, all these models were used along with multiple Balancing Techniques [9].

Alerts play important roles in such scenarios like threats, as when organizations and personnel gets consistent alerts and news from a source, they investigate it and try to think about their own integrity and security. for which we have integrated Telegram Channel on which autonomously an alert regarding leaks will be shared to our clients [10].

CHAPTER 3

DESIGN AND METHODOLOGY

3.1 Development Method

Agile Software Development Life Cycle (SDLC) has been utilized as the development method to develop *ThreatLens*. Agile was utilized due to its nature to be iterative and very effective in projects with high chances that their requirements are bound to change as time progresses. Because ThreatLens had so many elements embedded within it like website classification, dataset creation, model training, Telegram integration the team was able to build the system piece by piece, one might say and be receptive to comment and inspiration in the meantime.

3.1.1 Why Agile?

Agile was utilized over more conventional methodologies like the Waterfall model because it supported early release of key functionalities and modular construction.
For ThreatLens, with industry requirement and challenges, Agile allowed the team to re-prioritize and consolidate functionality from testing.

3.2 Tools and Technologies

The project uses multiple tools and technologies including web scraping frameworks, data processing, machine learning, libraries, Integration methods, storage solutions and communication APIs to create an automated hidden network threat intelligence system. These technology and tools selections were made keeping in mind reliability, performance and maintainability.

3.2.1 Programming Language

Python was used as the main language for all the backend operations due to its extensive library ecosystem and modern support for machine learning, data preprocessing and web scraping. Flask framework support for web application.

3.2.2 Web Application Framework

Flask was used as frontend framework for web integration with our python-based backend. Due to its flexibility minimal design. Also, it has some clean URL routing and HTTP responses.

3.2.3 Web Scrapping and Crawling Technologies

The following web scraping techniques were used in ThreatLens

3.2.3.1 Playwright

Playwright was used for browser automation of JavaScript rendered hidden network sites as it is the most suitable and reliable crawling technology so far supports headless

operations and handles dynamically rendered data so easily. Its configurable timeout and retry mechanisms make it a clear choice [11].

Use Cases:

- Crawling JavaScript-heavy sites
- For content which gets loaded dynamically on the site
- Handling multi page data extraction

3.2.3.2 REQUESTS

REQUESTS is an HTTP library used for content extraction from static and simple sites. It's a lightweight and fast method it has good session management capabilities. lower resource consumption as compared to other browser automation for simple pages [12].

Use Cases:

- Retrieving static HTML pages from hidden network sites
- Lightweight monitoring of leaks sites with minimal JavaScript
- Useful where auto retry and timeout required

3.2.3.3 Beautiful Soup

Beautiful Soup was used for parsing HTML and data extraction from hidden network sites. Beautiful Soup is a robust HTML handler, it navigates so smoothly through DOM structures, works excellent for CSS selectors and tag-based searching and extraction [13].

Use Cases:

- Parsing website content and metadata
- Extracting whole listing from sites right away
- Finding anything specific from the whole DOM in one go

3.2.4 Anonymous Network Access

The following Technologies were used for hidden network access and connections as per the requirement and practice of ***Genesis Technologies***:

3.2.4.1 TOR (The Onion Router)

TOR is a privacy-focused network which keeps you anonymous over the internet. TOR keeps your IP, your location, your identity, your internet activity all hidden. It's called onion because it is wrapped in multiple layers of encryption, like onion layers. We have used TOR SOCKS proxy. We have used 9050 proxy from SOCKS [14].

3.2.4.2 PySocks

PySocks is a python library used to add SOCKS proxy support which diverts traffic through a particular channel instead of sending it directly.it allows python programs to connect to the internet through a proxy server instead of directly.

3.2.5 Data Processing and Analysis

The following Data Processing and Analysis tools and techniques were used:

3.2.5.1 Pandas

Pandas were used for data manipulation and analysis as it's a strong python library. Pandas makes it way too easy to work with structured data. Pandas is used in various areas like data cleaning, data preprocessing, Exploratory data analysis (EDA) and machine learning pipeline. Works great with CSV, JSON and integration with visualization libraries.

3.2.5.2 NumPy

NumPy is a fundamental python library used for handling numerical arrays, fast mathematical computations it is almost the backbone of the whole data science and machine learning libraries including Pandas, Scikit-learn, TensorFlow, PyTorch, etc.

3.2.5.3 Regular Expression (re module)

The re module is used in python for working with regular expressions, usually used for matching, searching and manipulating text using patterns. A regex is a sequence of characters that forms a search pattern. It can be used in like extracting all the email addresses or IP addresses, parsing vulnerability identifiers etc [15].

3.2.5.4 Natural Language Processing

The following NLP methods were used in this project:

3.2.5.5 SpaCy

SpacCy is an advanced natural language processing and named entity recognition having multiple pretrained models for multiple languages with excellent NER (name entity recognition) can be used for custom NER.
We are using core SpaCy NLP pipeline which uses rule-based tagger and uses ***en_core_web_sm*** model [16].

3.2.5.6 NLTK (Natural Language Toolkit)

NLTK is used in ThreatLens for text processing and linguistic analysis because of its strong support for text classification tasks. It creates tokens, removes stop words, cleans and normalizes text[17].

3.2.6 Helping Tools used Throughout the Project

3.2.6.1 GitHub

GitHub was used to maintain the project and it's related files, which made managing this whole project very easy and trackable throughout the project.

3.2.6.2 GitHub Copilot

GitHub Copilot was used as a generative AI helper, which made some tedious task easier though its context aware code generation.

3.2.6.3 NotebookLM

NotebookLM was used as a research helper tool for studying Journals and extracting key insights from them.

3.3 System Architecture

The system architecture is specially designed as per the requirements of the industry as they needed a Website Classification model which will tell them if a particular site or a set of sites is important for them or not, and second thing they wanted was an AI model which will predict on the data they already have that what kind of industries can face data breaches in upcoming time.

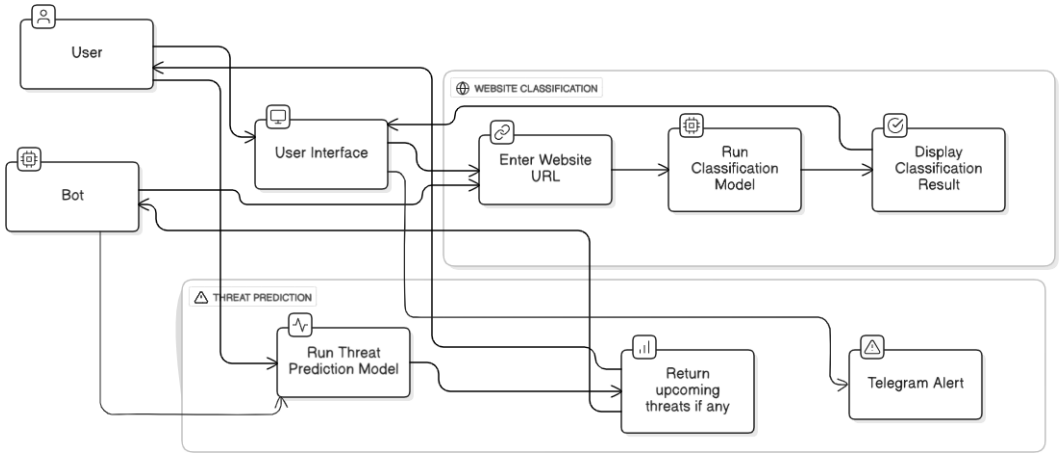


Figure 3.1 System Architecture

Now first let's talk about the architecture of Website Classification model in this we give URL to a site, it will access the site and in response to which it will tell the user what this site is related to. Flow is drawn in the diagram below Figure 3.2

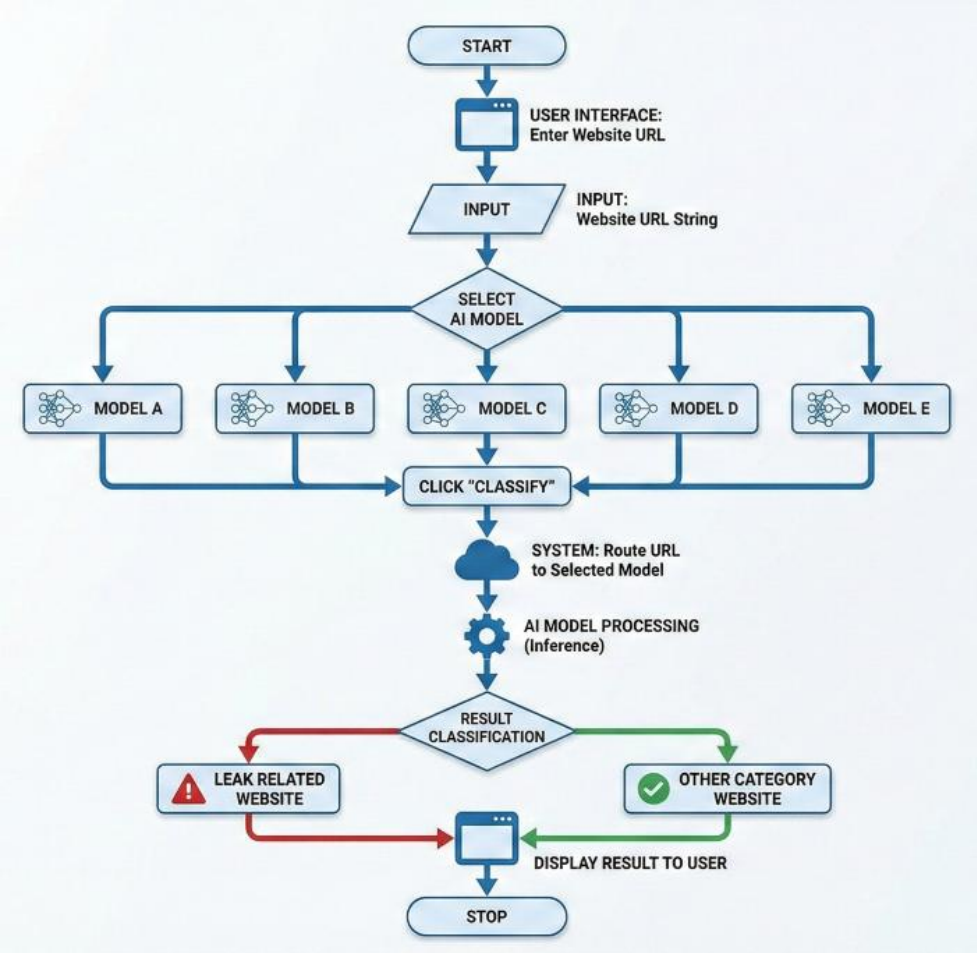


Figure 3.2 Website Classification Model

3.4 Use case Diagram

This use case diagram provides an overall view of interaction of users and the system of the *Orion Intelligence.* The automated calls to the models is shown as BOT in the use case diagram.

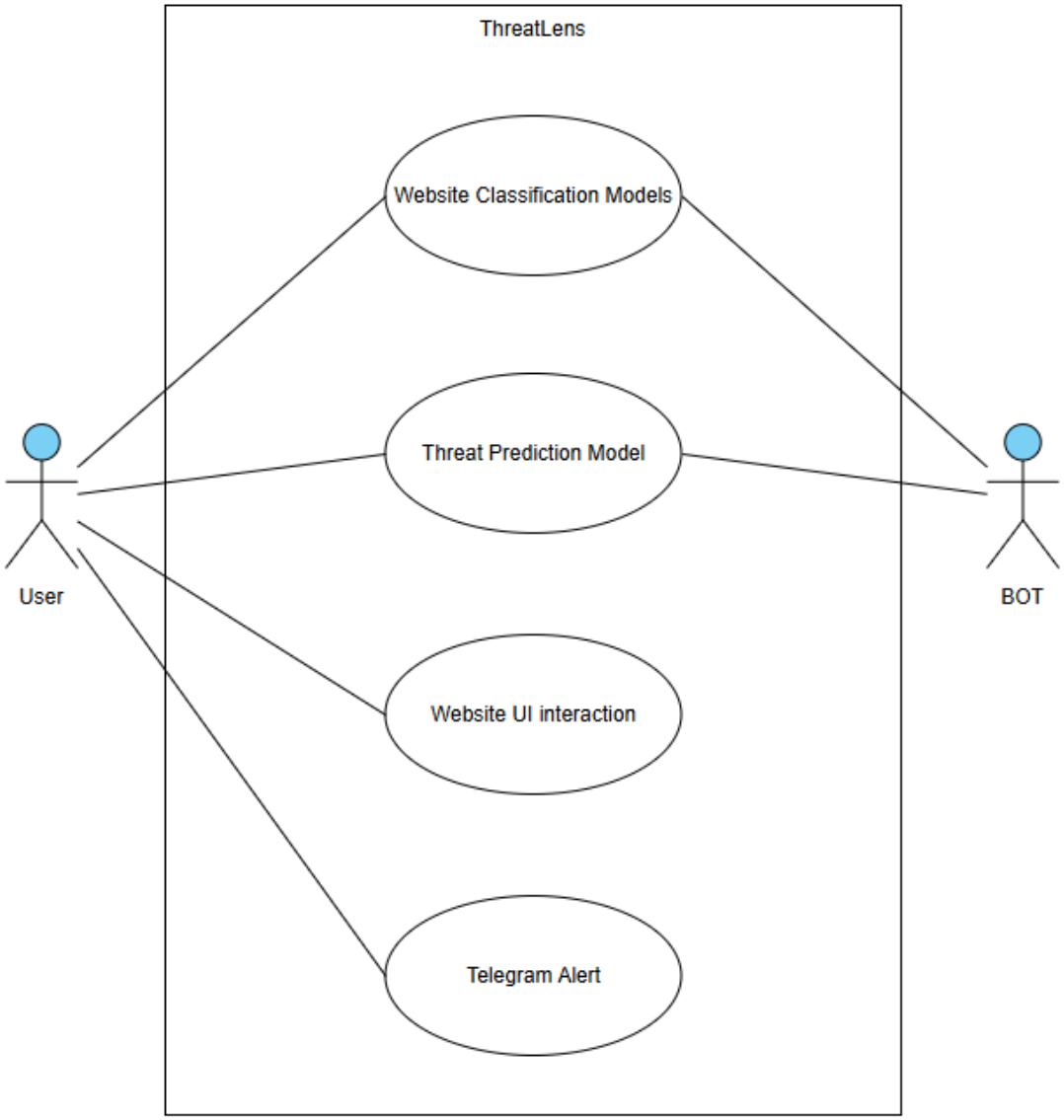


Figure 3.3 Use Case Diagram

3.4.1 Use Cases

Table 3.1 use case 1

USE CASE ID	**UC-1**
USE CASE NAME	**Website Classification Model**
Actors	**User / BOT**
Description	This model will allow user/bot to check the sites if they are of any use for the company or not
Trigger	User will be using it thorough the *ThreatLens/Orion Intelligence* UI
Preconditions	Sites should be either individual URL or in form of csv/excel format, and user/bot needs to select an model among 5 to get classification
Post conditions	In case of batch, the sheet must have a column of URLs named URL, URLs, sites to continue the classification
Normal Flow	1. Open the UI 2. Enter single or batch of URLs 3. Select TOR proxy 4. Get output either on UI if single URL or in downloadable file if batch of URLs
Alternative Flow	N/A
Exceptions	N/A
Includes	N/A
Special Requirement	N/A
Assumptions	User must understand English language
Notes and Issues	N/A

Table 3.2 use case 2

USE CASE ID	**UC-2**
USE CASE NAME	**Threat Prediction Model**
Actors	User/bot
Description	This model predicts what industry in upcoming time can get affected
Trigger	User/bot runs this model as per needs of the main project
Preconditions	Model Must be integrated with *Orion Intelligence*
Post conditions	N/A
Normal Flow	1. Trigger the model 2. Model runs 3. Model predicts
Alternative Flow	N/A
Exceptions	N/A
Includes	N/A
Special Requirement	N/A
Assumptions	N/A
Notes and Issues	N/A

CHAPTER 4

DATA AND EXPERIMENTS

4.1 Hidden Network Websites Classification

This is the phase from which we started the whole project as website classification and specifically for hidden networks sites it's a bit different from Clearnet sites. Because of the diversification of data present on the sites which usually Clearnet sites don't contain, sometimes we encounter data without any proper format or HTML structure pattern so we needed an AI Model for Hidden Network sites classification which should be trained on the data of hidden networks sites so that it may find patterns well and classify sites properly. For this reason, we have created a binary classification AI model which will predict either if the site is related to leaks or not.

Following are the procedures, steps and methodologies we adopted in this process.

4.1.1 Data Sources and initial steps

Finding hidden network was a tedious task but with the help of ***Genesis Technologies*** We were able to get websites from them and started to analyse and started to work on it. We scrapped these sites for keywords extraction with the help of which at the end we have classified sites, as different kind of sites contain words, sentences according to their niche and the service they provide. As a whole 200+ sites were used. We scraped sites using beautiful soup, REQUESTS, regular expressions, NLTK, and other

tools and technologies. We have used multiple AI models along with multiple Class Imbalance techniques for comparative analysis of all the models and techniques.

4.1.2 Dataset Structure

The dataset we created for the purpose of website classification consists of URLs of sites, 10 keywords from each site along with its frequency or occurrence, and finally the class label which is named as category in this scenario (either its leak or others).

4.1.3 Data Preprocessing Pipeline

Our preprocessing pipeline applies several critical steps to ensure a good and robust dataset.

4.1.3.1 Column standardization

All columns converted to lowercase and stripped of whitespaces to ensure consistency in column referencing throughout the pipeline.

4.1.3.2 Noise Removal

URL column was removed to prevent model from learning source-specific patterns. As it will not be providing any value to the model, instead it will cause some overfitting which can be scary for the model.

4.1.3.3 Data Shuffling

Data needed a shuffle because the sites were in a flow like in the start we had all sites related to leaks and at the bottom we had sites related to our second class which is others, Dataset is randomly shuffled with a fixed seed of 42. in this way we are preventing ordering bias in the training process.

4.1.3.4 Missing Values Handling

Our system is equipped with an intelligent missing value Handler which is category-aware, means missing values are filled using the mode (most frequent value) within the same category. This preserves the semantic relationship between keywords and threat categories. Ensures that imputed values are contextually sound. The system reports missing values before and after the imputation. Which ultimately provides transparency and data quality improvements.

4.1.3.5 Feature Engineering

Following feature Engineering methods were used:

4.1.3.5.1 Keyword Consolidation

All keyword columns are concatenated into a single text feature. Null values are excluded from concatenation as Vectorizers understand single column text lines as compared to single keyword in multiple columns.

4.1.3.5.2 Label Encoding

Categorical labels are converted into numerical format. Which makes dataset more compatible with the machine learning algorithms. Also label encoder is saved for inverse transformation during prediction [18].

4.1.3.5.3 TF-IDF Vectorization

Converts text keywords into numerical vectors with the help of this vectorizer we were able to captures the contextual relationships through bigram and produce sparse matrices for memory efficiency.

4.1.3.6 Class Imbalance Handling

Our system addresses class imbalance through two resampling techniques:

4.1.3.6.1 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generates synthetic samples for minority classes, creates new instances by interpolating between existing minority class samples. Uses K-nearest neighbours to identify similar samples. It helps reducing overfitting as compared to simple duplication, creates diverse synthetic samples [19].

4.1.3.6.2 Random Over-sampling

Random Over-sampling duplicates existing minority class samples randomly and balances class distribution by replication. This method is comparatively simple and computationally efficient, moreover it preserves original data distribution [20] .

4.1.4 Machine Learning Models

Five distinct classification algorithms were selected to evaluate different AI models for such kinds of problems.

4.1.4.1.1 Random Forest Classifier

Random forest classifier was selected as an ensemble method in our experimental framework due to its efficiency in handling text features and its robust performance across various classification tasks. Additionally random forest requires minimal hyperparameter tuning to achieve competitive baseline performance, which made it an excellent starting point for model comparison [21].

4.1.4.1.2 XGBoost

XGBoost is a state-of-the-art gradient boosting algorithm, efficient in handling sparse inputs, widely used in industry due to its high accuracy and flexibility. Gradient boosting of decision trees with advanced optimizations and regularization (L1/L2).
XGBoost supports multi-threading and supports early stopping. Although needs careful tuning and needs longer training time [22].

4.1.4.1.3 Multinomial Naïve Bayes

Naive Bayes is considered as a classic baseline for text classification in the AI industry, due to its extremely fast and memory efficient properties, it was the core reason behind considering it in this project. Works great with word frequency (TF-IDF or counts). Naïve bayes also considers conditional independence of given feature class [23].

4.1.4.1.4 CatBoost

CatBoost was chosen due to its modern gradient boosting and great out of the box performance lastly, it's robust handling over overfitting. Often requires less training than XGBoost. Have lower memory footprints and have built in handling for categorical data. CatBoost is also robust for Imbalance behaviour of Datasets [24].

4.1.4.1.5 Stochastic Gradient Boosting

Stochastic Gradient is a well-known classical baseline model to compare against modern varieties of Boosting algorithms (XGBoost, CatBoost). uses no extra dependencies beyond scikit-learn, easy to reproduce results. Although it has slower training speed and less optimized outcomes as compared to XGBoost/CatBoost [25].

4.2 The Threat Prediction

This is the main part of the project which needed us to think and code like a *Data scientist*. This part required extensive data preprocessing pipelines, unlimited changes and revisions. Because as a Data Scientist we need to extract key information and don't know directly if it will fit in or not, and especially when you are working on data from hidden networks.

And while doing all this we needed to figure out how to avoid the ***Curse of dimensionality*** [26]. Below are the steps and process we did while creating our dataset for this kind of purpose.

4.2.1 The Custom Scripts

Custom scripts are those scripts which we write for specific websites which we need to tailor as per the structure of the target site, to ensure that we are getting the correct and desired data from the target site. We use all the tools and technologies mentioned in the tools and technologies section above.

These custom scripts are mandatory for us, as we are talking about hidden networks sites and these sites have one of its own kind of structure of sites. So, any general scrapper can get failed here due to this issue, however we have used 20+ custom scripts in our project to ensure good data size we have for our model training.

4.2.2 Helper Methods

As we must extract data from multiple sites and we know very well we need to apply some sort of processes to make the data useful, so in this regard we have created multiple helper methods which will be getting input and will return useful information to use from the given input. Like when we give a *m_content* (an Enum in which we

are storing description of the data leak/breach) it will use some rule-based NLP or transformers model to get the keywords regarding industry or the departments that have been breached. Which will be ultimately used for our model training.

Following are the helper methods we have used in this project so far:

4.2.2.1 _breach_dept.py

SpaCy was used for tokenization, text processing and matching. Along with that we have used *PhraseMatcher* which is a rule-based phrase matching engine powered by SpaCy itself. Our helper method is well aware of contextual meanings like if it finds words like debit card, transection, bank etc it will identify that content that which department of the company was affected.

A sample chunk of code is given below:

Table 4.1 Sample Code Snippet for _breach_dept.py

```
# Financial data indicators
    financial_keywords = [
        "credit card", "debit card", "payment", "financial", "bank", "transaction",
        "account number", "routing number", "swift", "Iban", "card number",
        "cvv", "pin", "balance", "statement", "wire transfer"
    ]
    if any(word in text for word in financial_keywords):
        inferred.extend(["Finance", "Customer Service"])
```

Rule-based NLP was preferred here instead of any AI Model itself because of one main core reason, which is we needed this to create ourselves a dataset which needs some sort of pattern and sequence if we had used any ai model it would have returned any random out to us and then during model training it would have been an

tsunami for this project due to which our company guided us through and instructed to go for *Rule-Based NLP* instead of any AI Models.

4.2.2.2 _industry_extractor.py

This helper method was created in order to extract the affected industry from the given *m_content* (an Enum in which we are storing description of the data leak/breach) and store the output that weather it's an automobile industry, tech industry or any other. In this helper method we are using *Transformers (hugging face)* to perform *zero-shot classification.* It is a kind of thing which can match text to those categories even not by getting trained on them, this way working is called *Zero-Shot learning*.

The AI model predicts the industry from the given list of industries to it, to maintain a decorum and pattern throughout the dataset. Below is a code snippet for reference.

Table 4.2 Code Snippet for _industry_extractor.py

```
self.classifier = pipeline (
"Zero-shot-classification",
model="facebook/bart-large-mnli" )
```

4.2.2.3 _country_extractor.py

In this helper method we give it content, and it checks if there is any country name in that text or not, this specific helper method is used in scenarios where the country name of the company is not mentioned separately, so we need to check if there is country name mentioned in the description of the breach. It supports full country names and short forms of countries. If it does not find any country name, then it will return "USA" this is solution while we create datasets as at times we might not be able to find.

4.2.2.4 _URL_extractot.py

In this helper method we are simply parsing the content and matching if it gets a text like URL, we have created a Regex pattern for URLs (http, https, www, common TLDs, onion) which efficiently parses any URLs within the given content and store them accordingly.

4.2.2.5 _date_Normalizer.py

This is one of the most important helper method as there are a lot of time formats and getting these as it is can be a mess, so that is why we created this helper method which not only make it to one standard time format “DD/MM/YYYY” but also our helper is efficient enough to handle relative dates.

By relative dates, we are talking about things we have seen like “2 days ago” now in a dataset what is 2 days ago we don’t know, so in scenarios like this our helper method checks the current date on the go and stores the date of two days earlier in the record.

4.2.2.6 _string_cleaner.py

This helper method came into effect because when we extract data from internet, we get some unwanted things along with it like ‘\n’ etc which produces noise in our data so by passing through this function we are getting ourselves some neat and clean data,

4.2.3 Dataset Creation or Future expansion Pipeline

ThreatLens is designed in a way during its development keeping in mind the ease of any future expansion according to the requirements of the *Orion Intelligence* and the industry.

We have created a dynamic pipeline for this which will add up new data as per the format of current dataset with the help of a menu-based system scrapper, which will get the data as per the requirements of specific sites data, as data on site of hidden network are of so diverse nature. We have automated all the outsider helper and inhouse functions within the script which with help of each other creates and impactful dataset, which is ready to use, all preprocessing and checks are performed on the go. Which makes this an efficient and dynamic approach compatible with all custom scripts.

4.2.4 Data Preprocessing for Threat Prediction Models

4.2.4.1 Basic sanity checks

In this part we made sure that we don't have anything that have been missed while creation of the dataset, as there can be exceptions everywhere, so we performed basic dataset sanity checks like null values, data formats, etc which is mandatory for a dataset before applying any operations.

4.2.4.2 Feature Engineering

In this part we initially had our data in this form as shown in figure

date	country	industry	departments
14/11/2025	USA	Healthcare and Medical	Customer Service, Executive, Finance, HR, IT, Legal, Marketing, Medical, Sales
05/11/2025	Morocco, France	Retail and E-commerce	Customer Service, Finance, HR, IT, Legal, Manufacturing, Medical, Operations, Retail, Sales
05/11/2025	USA	Consulting and Professional Services	Customer Service, Finance, HR, R&D, Sales
29/10/2025	Spain, Portugal	Healthcare and Medical	Customer Service, Finance, HR, IT, Legal, Medical, Operations, Retail, Sales
22/10/2025	USA	Security Services	Customer Service, Finance, HR, IT, Operations, Retail, Sales
20/10/2025	Spain	Consulting and Professional Services	Customer Service, Finance, HR, IT, Manufacturing, Sales
20/10/2025	USA	Technology and Software	Customer Service, Data Analytics, Executive, Finance, HR, IT, Legal, R&D, Sales
20/10/2025	USA	Consulting and Professional Services	Customer Service, Finance, Government, IT, Legal, Medical

Figure 4.1 screenshot of raw dataset

As we know that we need to preprocess our data and apply feature engineering on the dataset before we can move towards model training on that dataset.

Here are some main steps we performed on the dataset:

- Removing null dates
- Removing country names more than one
- Made sure that every cell has got the correct data through our pipeline
- And a few other basic sanity checks

After applying all above steps, we moved then towards the feature engineering and extracted some new useful information which ultimately helped our model to learn features and return a good output.

Our feature engineering resulted in the following new features:

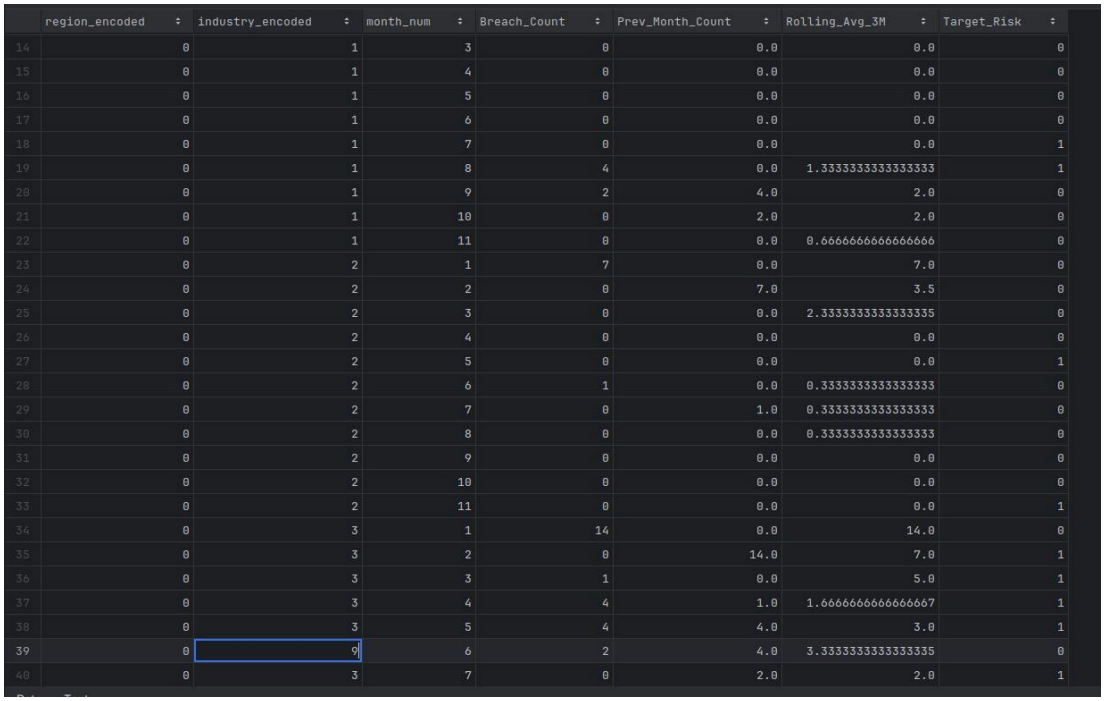
- Breach count
- Month number
- Previous month breach
- Rolling average of previous 3 months
- Target risk

So, from dataset shown in figure 4.1 above we extracted all these features and then moved towards our next phase.

4.2.4.3 Label Encoding

Label Encoding is used in this dataset for labelling industry, and the region features as they were treated as categorical columns/features.

After label encoding dataset looks like this:



	region_encoded	industry_encoded	month_num	Breach_Count	Prev_Month_Count	Rolling_Avg_3M	Target_Risk
14	0	1	3	0	0.0	0.0	0
15	0	1	4	0	0.0	0.0	0
16	0	1	5	0	0.0	0.0	0
17	0	1	6	0	0.0	0.0	0
18	0	1	7	0	0.0	0.0	1
19	0	1	8	4	0.0	1.3333333333333333	1
20	0	1	9	2	4.0	2.0	0
21	0	1	10	0	2.0	2.0	0
22	0	1	11	0	0.0	0.6666666666666666	0
23	0	2	1	7	0.0	7.0	0
24	0	2	2	0	7.0	3.5	0
25	0	2	3	0	0.0	2.3333333333333335	0
26	0	2	4	0	0.0	0.0	0
27	0	2	5	0	0.0	0.0	1
28	0	2	6	1	0.0	0.3333333333333333	0
29	0	2	7	0	1.0	0.3333333333333333	0
30	0	2	8	0	0.0	0.3333333333333333	0
31	0	2	9	0	0.0	0.0	0
32	0	2	10	0	0.0	0.0	0
33	0	2	11	0	0.0	0.0	1
34	0	3	1	14	0.0	14.0	0
35	0	3	2	0	14.0	7.0	1
36	0	3	3	1	0.0	5.0	1
37	0	3	4	4	1.0	1.6666666666666667	1
38	0	3	5	4	4.0	3.0	1
39	0	9	6	2	4.0	3.3333333333333335	0
40	0	3	7	0	2.0	2.0	1

Figure 4.2 picture of dataset after label encoding

4.3 The Ultimate ThreatLens Pipeline

This pipeline has multiple things working together, forming a strong and robust scrapper which creates a data input later fed to our AI models. The basic flow of the work is shown in figure 4.3 below.

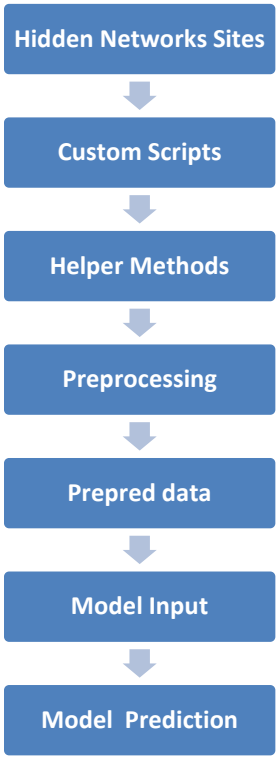


Figure 4.3 The ThreatLens Pipeline

This pipeline processes data throughout and make it in a useful way that can be used to run on the data coming from sites and resources of *Orion intelligence*.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Results of Website Classification Models

All the models performed somehow near to each other, below is the detailed discussion of all models with their performance metrices and comparative table at the end.

5.1.1 Naïve Bayes (The "Safety-First" Model)

Although it is not the most accurate in general terms, Naive Bayes exhibited a special property that is of utmost importance to safety systems: Zero False Negatives.

Optimal Strategy: SMOTE or Random Oversampling (Same results).

Key Metrics:

- Recall (Leaks): 100%
- Accuracy: 77.55%
- False Negatives: 0 (It failed to detect any leak)
- False Positives: 11 (High false alarm rate)

Balancing Technique: SMOTE

Accuracy: 0.7755

Table 5.1 Naive Bayed Performance using SMOTE

Class	Precision	Recall	F1-Score	Support
leak	0.6333	1.0000	0.7755	19
other	1.0000	0.6333	0.7755	30
Macro Avg	0.8167	0.8167	0.7755	49
Weighted Avg	0.8578	0.7755	0.7755	49

Confusion Matrix:

Table 5.2 Confusion Matrix for Naive Bayes using SMOTE

	Pred 0	Pred 1
True 0	19	0
True 1	11	19

Balancing Technique: Random Oversampling

Accuracy: 0.7755

Table 5.3 Naive Bayes Performance using Random Oversampling

Class	Precision	Recall	F1-Score	Support
leak	0.6333	1.0000	0.7755	19
other	1.0000	0.6333	0.7755	30
Macro Avg	0.8167	0.8167	0.7755	49
Weighted Avg	0.8578	0.7755	0.7755	49

Confusion Matrix:

Table 5.4 Confusion Matrix for Naive Bayes using Random Oversampling

	Pred 0	**Pred 1**
True 0	19	0
True 1	11	19

Analysis:

Naive Bayes was a paranoic classifier. It was more focused on finding all possible threats and this leads to a perfect Recall (1.0). Nevertheless, such aggressiveness had serious effects on Precision (63%), i.e. discrete non-leaks were often mistaken as leaks.

This model is the best candidate when the price of a missed leakage is disastrous (e.g., loss of life) and the price of a false alarm is insignificant. But in our case, we don't have such scenario so we wouldn't be preferring it.

5.1.2 Gradient Boosting (The "Runner-Up")

Gradient Boosting was the second-best showing high stability and high consistency no matter what balancing technique is applied.

Best Configuration: SMOTE or Random Oversampling (Results were identical).

Key Metrics:

- Accuracy: 79.59%
- Recall (Leaks): 78.9%
- Precision (Leaks): 71.4%

Balancing Technique: SMOTE

Accuracy: 0.7959

Table 5.5 Gradient Boosting Performance using SMOTE

Class	Precision	Recall	F1-Score	Support
leak	0.7143	0.7895	0.7500	19
other	0.8571	0.8000	0.8276	30
Macro Avg	0.7857	0.7947	0.7888	49
Weighted Avg	0.8017	0.7959	0.7975	49

Confusion Matrix:

Table 5.6 Confusion Matrix for Gradient Boosting using SMOTE

	Pred 0	Pred 1
True 0	15	4
True 1	6	24

Balancing Technique: Random Oversampling

Accuracy: 0.7959

Table 5.7 Gradient Boosting Performance using Random Oversampling

Class	Precision	Recall	F1-Score	Support
leak	0.7143	0.7895	0.7500	19
other	0.8571	0.8000	0.8276	30
Macro Avg	0.7857	0.7947	0.7888	49
Weighted Avg	0.8017	0.7959	0.7975	49

Confusion Matrix:

Table 5.8 Confusion Matrix for Gradient Boosting using Random Oversampling

	Pred 0	**Pred 1**
True 0	15	4
True 1	6	24

Analysis:

Gradient Boosting was found to be more successful than the two algorithms (Random Forest and XGBoost) in terms of general accuracy. It had good balance with the loss of 4 leaks (False Negatives) and produced 6 false alarms (False Positives). It was found to be a strong competitor to CatBoost but a bit less precise during the last validation.

It can be a strong backup model; it offers consistent performance and is less sensitive to the specific type of data balancing used compared to other models.

5.1.3 XGBoost (The Underperformer)

even though XGBoost is known to work well with the state-of-the-art algorithm, it did not work well with this dataset like others

Best Configuration: SMOTE (for Accuracy) or Random Oversampling (for Recall).

Key Metrics (Random OS):

- Accuracy: 75.51%
- Recall (Leaks): 78.9%
- False Negatives: 4

Balancing Technique: SMOTE

Accuracy: 0.7755

Table 5.9 XGBoost Performance using SMOTE

Class	Precision	Recall	F1-Score	Support
leak	0.7000	0.7368	0.7179	19
other	0.8276	0.8000	0.8136	30
Macro Avg	0.7638	0.7684	0.7658	49
Weighted Avg	0.7781	0.7755	0.7765	49

Confusion Matrix:

Table 5.10 Confusion Matrix for XGBoost using SMOTE

	Pred 0	Pred 1
True 0	14	5
True 1	6	24

Balancing Technique: Random Oversampling

Accuracy: 0.7551

Table 5.11 XGBoost Performance using Random Oversampling

Class	Precision	Recall	F1-Score	Support
leak	0.6522	0.7895	0.7143	19
other	0.8462	0.7333	0.7857	30
Macro Avg	0.7492	0.7614	0.7500	49
Weighted Avg	0.7709	0.7551	0.7580	49

Confusion Matrix:

Table 5.12 Confusion Matrix for XGBoost using Random Oversampling

	Pred 0	Pred 1
True 0	15	4
True 1	8	22

Analysis:

XGBoost with Random Oversampling gave a better result in locating leaks (Recall over 79%) with a lower overall accuracy of 75.5%. The accuracy was increased to 77.5 with the usage of SMOTE but the leak detection (Recall) was reduced to 73.6. It could not reach the sweet spot that *CatBoost* was able to reach.

Not recommended, it introduced more complexity without outperforming the simpler Gradient Boosting or the smarter CatBoost model.

5.1.4 Random Forest (The Baseline)

The baseline model was the random Forest. Although interpretable, it performed enormously poorly with the minority population (SMOTE Leaks) when training on minorities.

Best Configuration: Random Oversampling

Key Metrics (Random OS):

- Accuracy: 77.55%
- Recall (Leaks): 73.7%
- False Negatives: 5

Balancing Technique: SMOTE

Accuracy: 0.7347

Table 5.13 Random Forest Performance using SMOTE

Class	Precision	Recall	F1-Score	Support
leak	0.6667	0.6316	0.6486	19
other	0.7742	0.8000	0.7869	30
Macro Avg	0.7204	0.7158	0.7178	49
Weighted Avg	0.7325	0.7347	0.7333	49

Confusion Matrix:

Table 5.14 Confusion Matrix for Random Forest using SMOTE

	Pred 0	Pred 1
True 0	12	7
True 1	6	24

Balancing Technique: Random Oversampling

Accuracy: 0.7755

Table 5.15 Random Forest Performance using Random Oversampling

Class	Precision	Recall	F1-Score	Support
leak	0.7000	0.7368	0.7179	19
other	0.8276	0.8000	0.8136	30
Macro Avg	0.7638	0.7684	0.7658	49
Weighted Avg	0.7781	0.7755	0.7765	49

Confusion Matrix:

Table 5.16 Confusion Matrix for Random Forest using Random Oversampling

	Pred 0	Pred 1
True 0	14	5
True 1	6	24

Analysis:

With SMOTE, the worst performance was demonstrated by the Random Forest, which had the lowest accuracy of the group (73.47), and 7 real leaks were missed. The Random Oversampling method raised its accuracy to a decent 77.55 percent, which is still lower than the boosting algorithms (CatBoost and Gradient Boosting), but is better than the base rate method used in classifying faint leak patterns.

Useful as a baseline for comparison, but insufficient for the final production system due to lower Recall compared to the boosting ensemble methods.

5.1.5 CatBoost (The "Primary Choice")

CatBoost was smarter and more balanced among the set of models and managed to address the trade-off between sensitivity (finding leaks) and specificity (false alarms).

Best Configuration: Random Oversampling.

Key Metrics (Random OS):

- Accuracy: 81.63% (Highest of all models tested)
- Recall (Leaks): 84.2%
- Precision (Leaks): 72.7%
- False Negatives: 3 (It missed only 3 leaks)
- False Positives: 6 (It raised 6 false alarms)

Balancing Technique: SMOTE

Accuracy: 0.7959

Table 5.17 CatBoost Performance using SMOTE

Class	Precision	Recall	F1-Score	Support
leak	0.7143	0.7895	0.7500	19
other	0.8571	0.8000	0.8276	30
Macro Avg	0.7857	0.7947	0.7888	49
Weighted Avg	0.8017	0.7959	0.7975	49

Confusion Matrix:

Table 5.18 Confusion Matrix for CatBoost using SMOTE

	Pred 0	Pred 1
True 0	15	4
True 1	6	24

Balancing Technique: Random Oversampling

Accuracy: 0.8163

Table 5.19 CatBoost Performance using Random Oversampling

Class	Precision	Recall	F1-Score	Support
leak	0.7273	0.8421	0.7805	19
other	0.8889	0.8000	0.8421	30
Macro Avg	0.8081	0.8211	0.8113	49
Weighted Avg	0.8262	0.8163	0.8182	49

Confusion Matrix:

Table 5.20 Confusion Matrix for CatBoost using Random Oversampling

	Pred 0	**Pred 1**
True 0	16	3
True 1	6	24

Analysis:

CatBoost was shown to be better than the other gradient boosting algorithms (XGBoost and vanilla Gradient Boosting) since it was successful in capturing complicated patterns in the minority class (Leaks).

The Sweet Spot: In comparison to Naive Bayes, which had a high recall rate by flagging everything as a leak, CatBoost found a balance that is smart. It had a high True Negative rate (24 correct other classifications) and was able to identify 16 out of 19 of the leaks.

Stability Check: The model used in the 5-fold cross-validation audit showed a mean accuracy of 80.48%. It was a bit volatile (8.9) because of the small size of the dataset,

but its highest performance was 89.58% in good data splits, which proves its great potential.

5.1.6 Comparison of all Models

Table 5.21 Comparison of all Models

Model	Balancing Technique	Accuracy	Precision	Recall	F1-Score
Random Forest	SMOTE	0.7347	0.7325	0.7347	0.7333
Random Forest	Random Oversampling	0.7755	0.7781	0.7755	0.7765
Naive Bayes	SMOTE	0.7755	0.8578	0.7755	0.7755
Naive Bayes	Random Oversampling	0.7755	0.8578	0.7755	0.7755
XGBoost	SMOTE	0.7755	0.7781	0.7755	0.7765
XGBoost	Random Oversampling	0.7551	0.7709	0.7551	0.7580
CatBoost	SMOTE	0.7959	0.8017	0.7959	0.7975
CatBoost	**Random Oversampling**	**0.8163**	**0.8262**	**0.8163**	**0.8182**
Gradient Boosting	SMOTE	0.7959	0.8017	0.7959	0.7975
Gradient Boosting	Random Oversampling	0.7959	0.8017	0.7959	0.7975

5.2 Results of Threat Prediction Model

5.2.1 Industry Prediction Model

This model of our predicts which industry in upcoming times can get affected, the model selection for this prediction was a challenging phase for us, we concluded on model chaining and after multiple meetings and discussions with the company we finalized these models:

- Random Forest
- XGBoost

As we have converted our dataset into time-series dataset, so we finalized these and decided we will check all of them for comparative analysis and experimentation. However, Random Forest outperformed all others clearly. Let's discuss them thoroughly.

5.2.1.1 Random Forest (The "primary choice")

Among all the models which were tested on this dataset, only random forest was able to discriminate threats and no threats, unlike others just to put everything at threat alert.

The main reason behind choosing Random Forest was its high recall as in cybersecurity we need high recall, it's more important than high accuracy. As let's say a model predicts an industry which is safe now, but our model predicts as if it can get affected, this is annoying but manageable, whereas on the other side if our model predicts any industry which is on threat as safe then that is a tsunami for the company. This is the core reason why we preferred Random Forest as this gave us high recall (90.52%) and accuracy 78.08% which is completely fine.

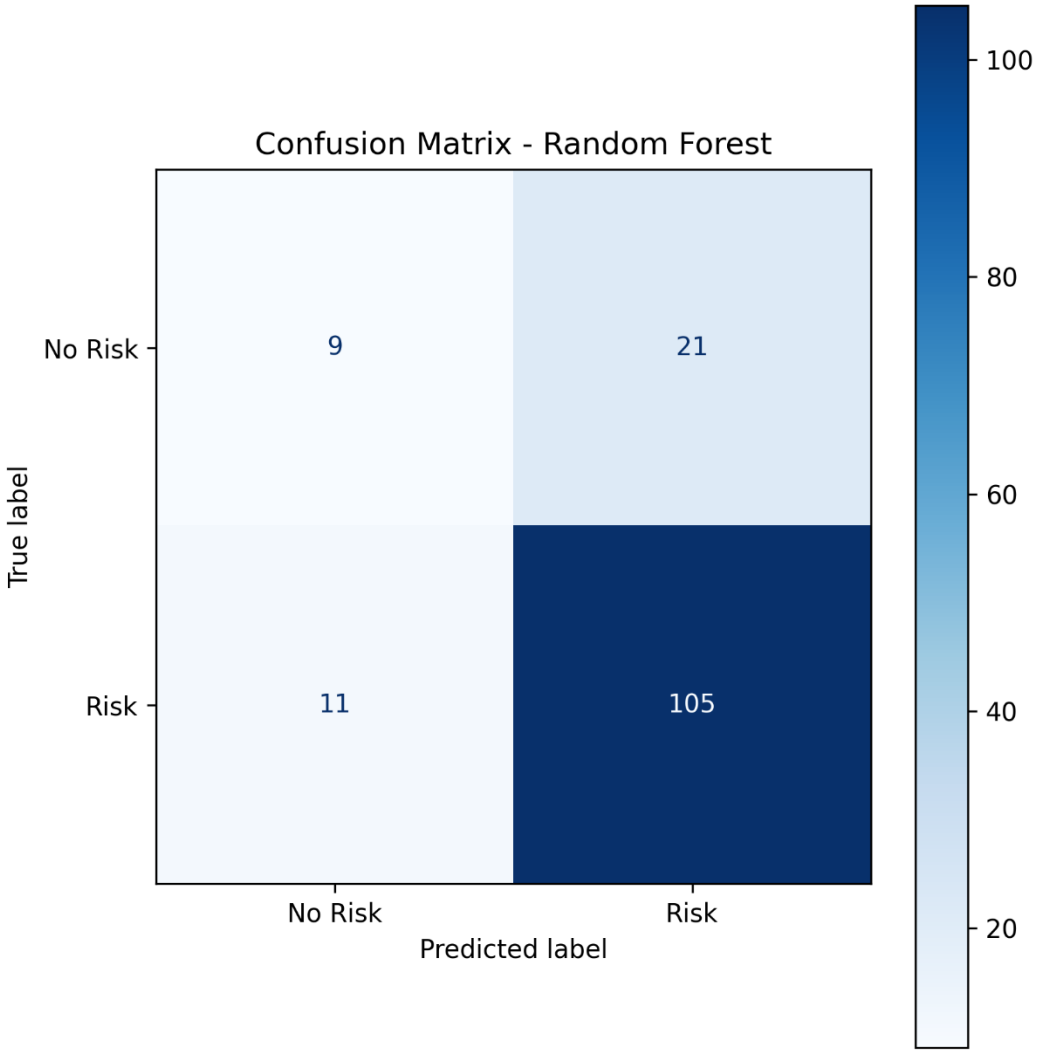


Figure 5.1 Confusion Matrix for Random Forest

Table 5.22 Performance Metrices for Random Forest

Performance Metrices for Random Forest	
Accuracy	78.08%
Recall	90.52%
Precision	83.33%
F1-Score	86.78%

5.2.1.2 XGBoost (High Precision Alternative)

This model gave us some decent results we got a good precision (83.93%), but unfortunately it missed 22 threats, which gave it the lowest recall. Good only if minimizing false alarm is important.

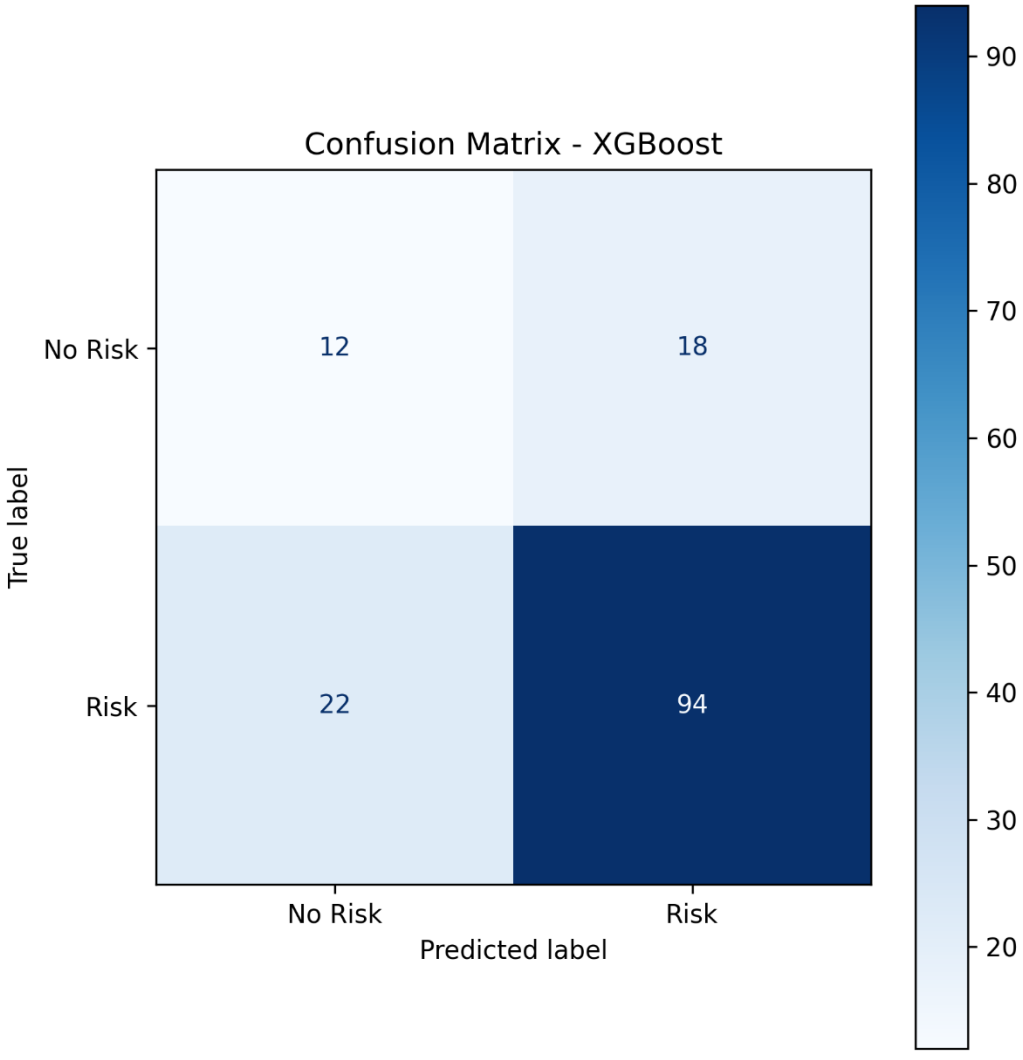


Figure 5.2 Confusion Matrix for XGBoost

Table 5.23 Performance metrices of XGBoost

Performance Metrices for XGBoost	
Accuracy	72.60 %
Recall	81.03 %
Precision	83.93 %
F1-Score	84.55 %

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

The ThreatLens project was able to achieve its main objective, which was to create an AI module that can analyse trends and patterns in hidden networks and help users of the Orion Intelligence platform to have proactive threat prediction. This development directly addresses the critical gap in existing security intelligence platforms which was previously limited to reactive solutions which failed to alert organisations while leaked data was actively shared on hidden networks.

The project methodology faced a lot of obstacles, including a lack of a public dataset suitable for the hidden network analysis. This necessitated the methodical construction of a proprietary dataset with specialized and anonymous web crawling techniques using tools such as Playwright and REQUESTS combined with an entire preprocessing pipeline containing specialized Rule-Based Natural Language Processing (NLP) using tools for extracting essential information.

The final ThreatLens system is composed of two primary AI components:

Website Classification Model: This binary classification model identifies if a hidden network site is relevant to leaks. After comparing five different algorithms, the CatBoost model, utilizing Random Oversampling, was selected as the Primary Choice. CatBoost was chosen because it was determined to be the smartest and most balanced model among the candidates, effectively navigating the trade-off between identifying true leaks (sensitivity) and minimizing false alarms (specificity)

Threat Prediction Model: This component forecasts which industries have a high chance of facing data breaches. The **Random Forest** algorithm was finalized as the preferred choice for this task. In the context of cybersecurity, the selection prioritized models that offer **high detection rates**, acknowledging that a missed threat could be catastrophic a "tsunami" for the company making high detection more important than simply having high general accuracy.

This industrial collaboration met the need for a leading-edge predictive security solution by incorporating ThreatLens directly into Orion Intelligence. Crucially, the system can keep clients up to date about the latest threats and breaches in an automated way by providing proactive alerts through a dedicated telegram channel, thus significantly boosting organizational cyber security preparedness.

6.2 Future Recommendations

As this is the very first step for the *Orion intelligence* for proactive cyber predictions, that is why we used to create specialized pipelines including Rule-Based Natural Language Processing (NLP), but in future we intend to move towards Advanced Natural Language Processing and deep learning models as the data grows.

In future Genesis technologies intend to have a fully automated Telegram BOT which will provide insight to it’s clients and keep the society aware of cyber threats and other dark web activities.

REFERENCES

[1] C. Secure, “The average time to indentify and contain a data breach is 280 days.” Accessed: Dec. 01, 2025. [Online]. Available: https://blog.camelsecure.com/camel-community-02152021

[2] “AI Driven Threat Intelligence and Predictive Cybersecurity Using Machine Learning to Forecast and Prevent Cyberattacks Before They Occur.”

[3] “Surface web,” *Wikipedia*. Nov. 12, 2025. Accessed: Dec. 01, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Surface_web&oldid=1321734244

[4] E. Burns and K. Buks, “AI-Driven Threat Intelligence and Predictive Cyber Defense”.

[5] “Crawling the Hidden Web (Stanford InfoLab).”

[6] O. David, “Automated Web Crawlers for Discovering Emerging Phishing Websites: A Threat Intelligence Approach”.

[7] V. Moka, “Natural Language Processing for Cyber Threat Intelligence Analysis”.

[8] S. Gupta, “Artificial Intelligence in Cyber Threat Detection: A Survey of Predictive Security Systems”.

[9] “Hybrid AI Models in Network Security: Combining ML, DL, and Rule-Based Systems,” *Int. J. Emerg. Res. Eng. Technol.*, vol. 5, 2024, doi: 10.63282/3050-922X.IJERET-V5I4P111.

[10] “Telegram bot for recognizing phishing links using machine learning.”

[11] “Difference between Playwright and Selenium,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/software-testing/difference-between-playwright-and-selenium/

[12] “Python Requests,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/python-requests-tutorial/

[13] “Implementing Web Scraping in Python with BeautifulSoup - GeeksforGeeks.” Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/implementing-web-scraping-python-beautiful-soup/

[14] “Tor Browser - A Complete Overview - GeeksforGeeks.” Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/techtips/tor-browser-a-complete-overview/

[15] “Python RegEx - GeeksforGeeks.” Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/python-regex/

[16] “spaCy for Natural Language Processing - GeeksforGeeks.” Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/nlp/spacy-for-natural-language-processing/

[17] “NLTK - NLP - GeeksforGeeks.” Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/nltk-nlp/

[18] “Label Encoding in Python,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/ml-label-encoding-of-datasets-in-python/

[19] “ML | Handling Imbalanced Data with SMOTE and Near Miss Algorithm in Python,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/

[20] “RandomOverSampler — Version 0.14.0.” Accessed: Nov. 29, 2025. [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html

[21] “Random Forest Algorithm in Machine Learning,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/

[22] “XGBoost,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/xgboost/

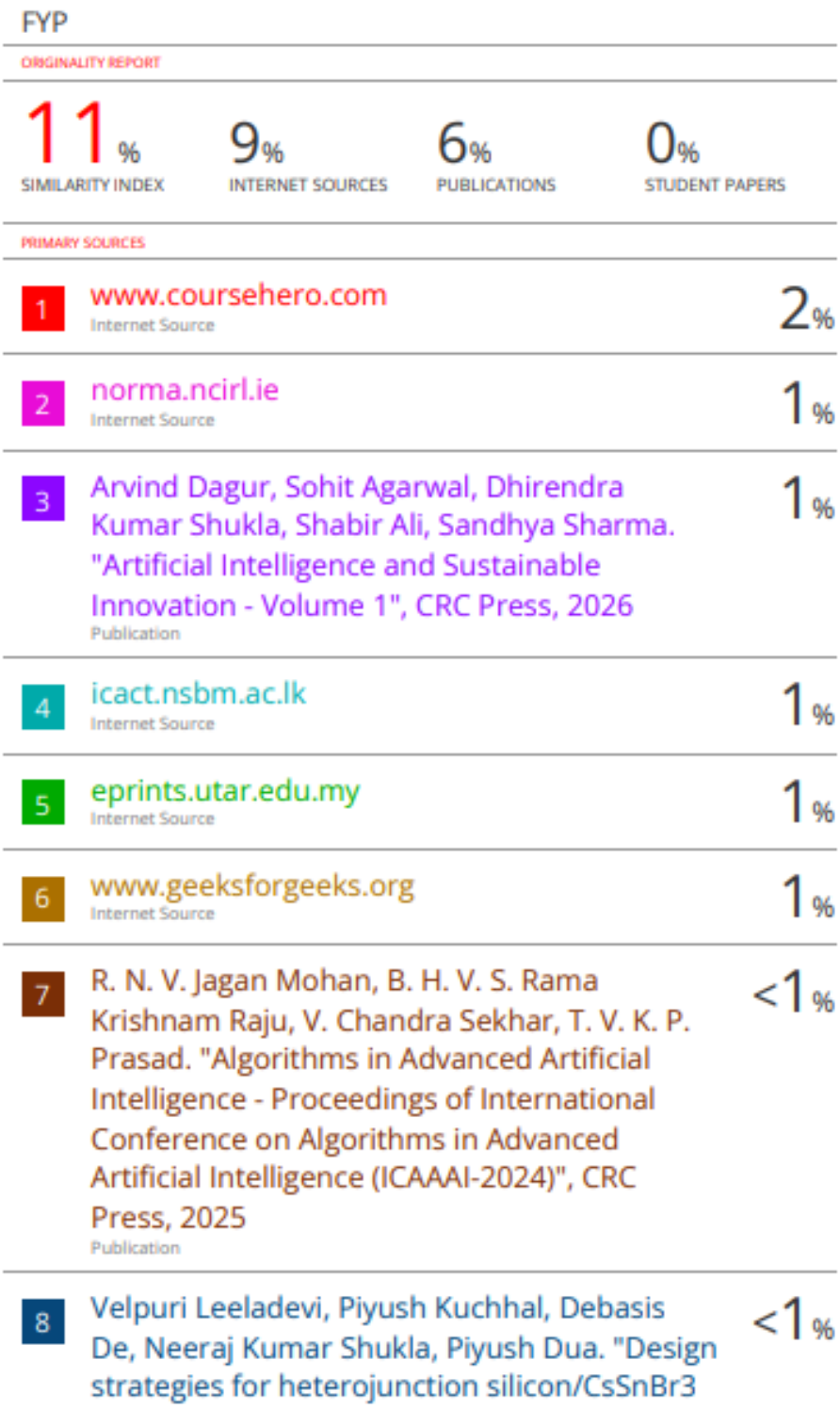
[23] “Multinomial Naive Bayes,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/multinomial-naive-bayes/

[24] “CatBoost in Machine Learning - GeeksforGeeks.” Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/catboost-ml/

[25] “Gradient Boosting in ML,” GeeksforGeeks. Accessed: Nov. 29, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/ml-gradient-boosting/

[26] “Curse of Dimensionality in Machine Learning,” GeeksforGeeks. Accessed: Nov. 28, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/curse-of-dimensionality-in-machine-learning/

Appendix



FYP

ORIGINALITY REPORT

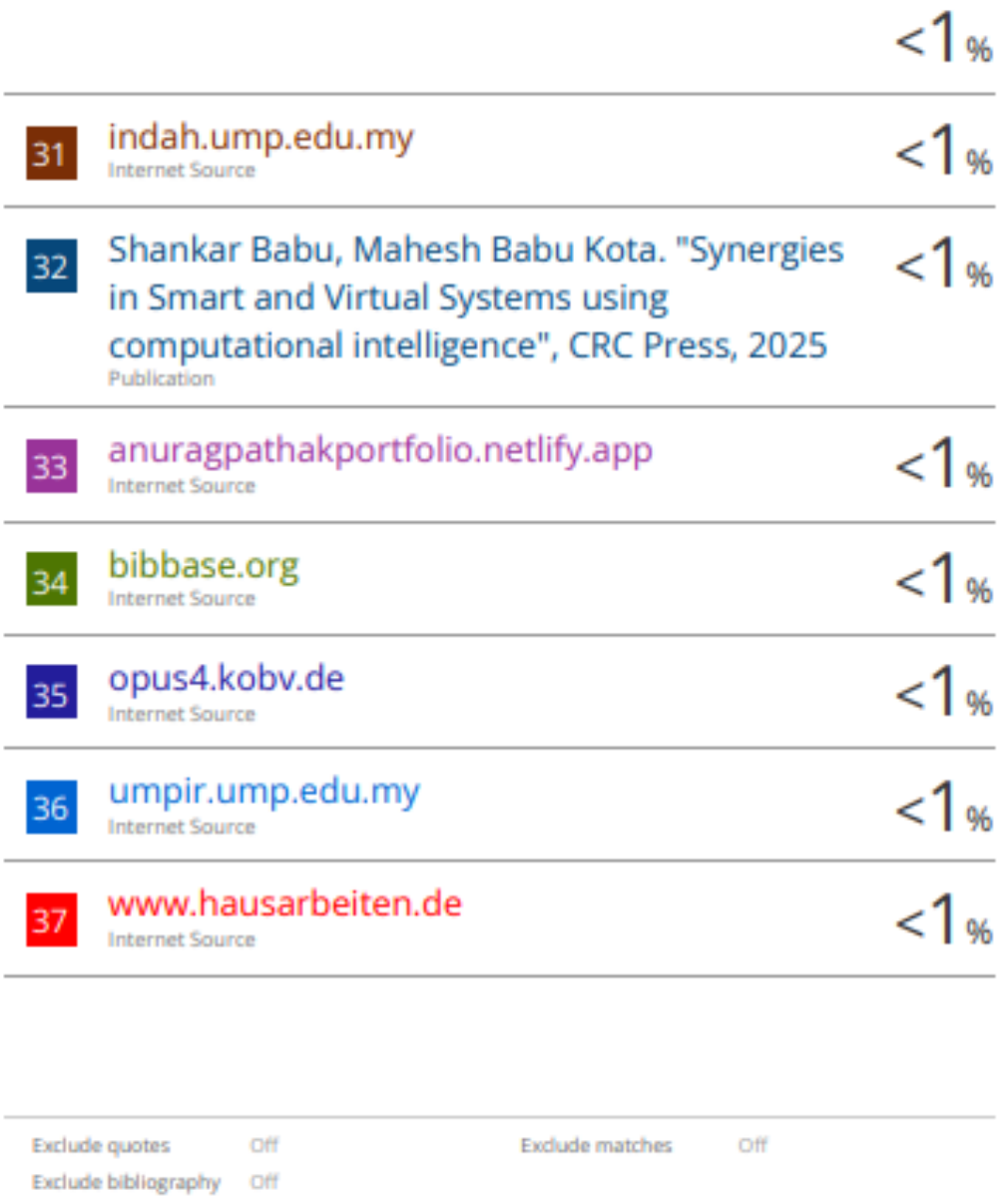
11%	9%	6%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	norma.ncirl.ie Internet Source	1%
3	Arvind Dagur, Sohit Agarwal, Dhirendra Kumar Shukla, Shabir Ali, Sandhya Sharma. "Artificial Intelligence and Sustainable Innovation - Volume 1", CRC Press, 2026 Publication	1%
4	icact.nsbm.ac.lk Internet Source	1%
5	eprints.utar.edu.my Internet Source	1%
6	www.geeksforgeeks.org Internet Source	1%
7	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P. Prasad. "Algorithms in Advanced Artificial Intelligence - Proceedings of International Conference on Algorithms in Advanced Artificial Intelligence (ICAAAI-2024)", CRC Press, 2025 Publication	<1%
8	Velpuri Leeladevi, Piyush Kuchhal, Debasis De, Neeraj Kumar Shukla, Piyush Dua. "Design strategies for heterojunction silicon/CsSnBr3	<1%

	lead-free tandem solar cells using machine learning and SCAPS-1D", Results in Engineering, 2025 Publication	
9	www.um.edu.mt Internet Source	<1 %
10	Khalifa Afane, Yijun Zhao. "Selecting Classifiers and Resampling Techniques for Imbalanced Datasets: A New Perspective", Procedia Computer Science, 2024 Publication	<1 %
11	ijeret.org Internet Source	<1 %
12	www.fpb.org.za Internet Source	<1 %
13	Debendra Muduli, Rahul Kumar Gupta, Samir Kumar Majhi, Binayak Ojha, Banshidhar Majhi. "An optimized learning approach for enhancing the security of digital twin-enabled industrial systems from distributed denial-of-service attacks", Journal of Industrial Information Integration, 2025 Publication	<1 %
14	www.mdpi.com Internet Source	<1 %
15	S.P. Jani, M. Adam Khan. "Applications of AI in Smart Technologies and Manufacturing", CRC Press, 2025 Publication	<1 %
16	arxiv.org Internet Source	<1 %
17	jurnal.yoctobrain.org Internet Source	

		<1%
18	blog.camelsecure.com Internet Source	<1%
19	desklib.com Internet Source	<1%
20	pdfcoffee.com Internet Source	<1%
21	Prasad, Akhilesh. "Forecasting spikes in the CBOE VIX Index", SP Jain School of Global Management (India), 2024 Publication	<1%
22	matjournals.net Internet Source	<1%
23	www.kcsfoundation.org Internet Source	<1%
24	Submitted to Higher Education Commission Pakistan Student Paper	<1%
25	ikee.lib.auth.gr Internet Source	<1%
26	research.thea.ie Internet Source	<1%
27	spectrum.library.concordia.ca Internet Source	<1%
28	www.guvi.in Internet Source	<1%
29	ir.jkuat.ac.ke Internet Source	<1%
30	dspace.atilim.edu.tr Internet Source	



		<1 %
31	indah.ump.edu.my Internet Source	<1 %
32	Shankar Babu, Mahesh Babu Kota. "Synergies in Smart and Virtual Systems using computational intelligence", CRC Press, 2025 Publication	<1 %
33	anuragpathakportfolio.netlify.app Internet Source	<1 %
34	bibbase.org Internet Source	<1 %
35	opus4.kobv.de Internet Source	<1 %
36	umpir.ump.edu.my Internet Source	<1 %
37	www.hausarbeiten.de Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		