An R-based System Implementation for Automated Threat Intelligence Analysis Integrating Text Mining and Machine Learning

Hung-Cheng Yang, I-Long Lin, Yu-Shan Lin, Chorng-Ming Chen*

Department of Computer Science and Information Engineering, Tatung University, Taiwan
yangyeh5046@gmail.com, cyberpaul@gm.ttu.edu.tw, hinet.lys6103@gmail.com, avenchentw@gmail.com

Abstract

The escalating threat of global cyberattacks targeting network systems and exfiltrating sensitive information has rendered traditional cybersecurity defense mechanisms increasingly inadequate. Consequently, assisting enterprises in constructing a comprehensive Cyber Threat Intelligence (CTI) aggregation and analysis framework has become critically important. This study aims to address a major pain point faced by cybersecurity analysts—namely, the significant time and resources required to manually process vast amounts of unstructured CTI reports—by proposing an innovative automated analytical solution.

The proposed research framework integrates the R programming language with Text Mining and Machine Learning (ML) techniques. Initially, CTI reports collected from institutions such as Taiwan's National Information Sharing and Analysis Center (N-ISAC) were processed through text mining. Using Term Frequency–Inverse Document Frequency (TF-IDF), we extracted lexical features and constructed a structured dataset. Subsequently, multiple machine learning classifiers were implemented and evaluated, including the C4.5 decision tree, Naive Bayes, Logistic Regression, and Classification and Regression Tree (CART) models, to automatically identify and categorize potential threats and vulnerabilities [1].

Experimental results demonstrated that the Naive Bayes classifier achieved the highest performance with an accuracy rate of **95.48%** on the CTI dataset. Moreover, this study successfully implemented a CTI analysis system equipped with visualization capabilities. Empirical validation confirmed that the system significantly reduces the time required for cybersecurity professionals to assess threat intelligence and rapidly generate remediation or hardening strategies for risk mitigation. The proposed research provides enterprises with a high-efficiency, high-accuracy, and scalable CTI analytical tool, effectively enhancing organizational cybersecurity resilience and forensic integrity within Security Information and Event Management (SIEM) and Managed Detection and Response (MDR) environments.

Keywords: Threat intelligence, Text mining, Feature selection, Machine learning

1 Introduction

The rapid development and widespread application of information technology have led to increasingly diverse and complex global cybersecurity threats, posing significant challenges to enterprises and organizations. Daily, cybersecurity professionals must process massive amounts of threat intelligence to protect information assets and sensitive data from unauthorized access or malicious attacks. In this demanding context, threat intelligence analysis and cybersecurity risk mitigation have become pivotal areas in information security.

This study aims to identify potential security threats and vulnerabilities through the analysis and detection of extensive intelligence data provided by regulatory authorities, thereby enabling the proactive prevention of cyberattacks. Cybersecurity risk management refers to the systematic methods and techniques employed to effectively reduce and eliminate risks, ensuring the confidentiality, integrity, and availability of systems and information assets [2].

To achieve this primary objective, this study utilizes the R programming language for data analysis and processing to develop a novel threat intelligence analysis model and a corresponding cybersecurity risk mitigation mechanism. The effectiveness and feasibility of the proposed methodologies will be rigorously evaluated through comprehensive system implementation and validation.

The adoption of R as the primary tool for data analysis leverages its well-established research applications, robust technical environment, and collaborative ecosystem, thereby ensuring continuity and methodological rigor. From a cybersecurity perspective, vulnerability reports released by the National Cybersecurity Agency over the past three years indicate no high-risk security patches associated with R, demonstrating its relative stability and reducing concerns regarding its practical application.

Furthermore, R is a mature language for data analysis and statistical modeling, enabling rapid data cleaning, transformation, and visualization. In cybersecurity threat intelligence analysis, R's efficient data processing packages (such as dplyr, tidyr, and data.table) significantly enhance research efficiency when dealing with diverse log, network traffic, and event data. Its open-source community continuously provides rich tools, including text mining (tm, tidytext), machine learning (caret, randomForest), and network analysis (igraph, ggraph), meeting the needs of

*Corresponding Author: I-Long Lin; Email: cyberpaul@gm.ttu.edu.tw

DOI: https://doi.org/10.70003/160792642026012701012

keyword extraction, pattern detection and attack behavior correlation analysis in threat intelligence research.

Finally, R excels in data visualization (e.g., ggplot2, shiny), allowing for the intuitive presentation of complex threat patterns and attack paths. This capability aids researchers and decision-makers in quickly grasping key insights, which is invaluable for cybersecurity intelligence sharing and threat awareness. In summary, the choice of the R programming language for this study's threat intelligence analysis and risk management framework is justified by its functional richness, suitability for algorithm development, stability, scalability, and strong community support, all of which ensure the high effectiveness and adaptability of the research outcomes in practical applications.

2 Literature Review

This section discusses the relevant literature concerning threat mapping and risk elimination, specifically focusing on research related to threat detection technologies and classifications. We review the main methodologies employed in these detection technologies and the challenges they currently face. Threat detection techniques can generally be categorized into three domains: textual analysis (or text mapping), machine learning, and example-based validation [3].

2.1 Threat Intelligence Sharing Standards and Guidelines

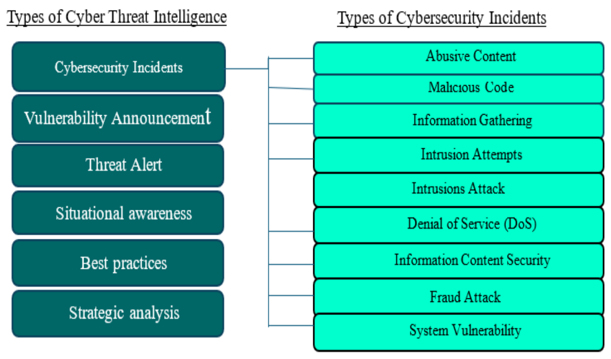


Figure 1. N-ISAC intelligence classification framework (source: **National Institute of Cyber Security**)

Threat intelligence is a complex, multi-dimensional domain upon which enterprises are increasingly reliant. Consequently, robust threat intelligence sharing mechanisms are crucial. **The Structured Threat Information eXpression (STIX)** is a standardized language specifically designed for describing, acquiring, standardizing, and communicating cyber threat intelligence. It features encapsulated intelligence, high readability, and extensibility, which collectively assist organizations in sharing and applying information security intelligence effectively. In accordance with the specification established by the **National Institute for Information Security (NII)**, the **JavaScript Object Notation (JSON)** data format is utilized as the information exchange language. Furthermore, six major information types are defined based on the STIX format. These security event types are further subdivided to ensure clear and granular identification. The specific information formats exchanged by the notification response subgroups have been clearly delineated by the National Information Security **Reporting and Notification Response Group (NISRG)**, as illustrated in Figure 1 [4-5].

2.2 Literature Mapping

Text mining is a cross-domain application that integrates techniques from data mining, natural language processing (NLP), and information retrieval to analyze large volumes of textual information and extract useful knowledge. Its primary applications span trend prediction, crime analysis, knowledge extraction, knowledge management, and email detection. In the context of this study, the emergence of text mining provides a viable solution to manage the massive volume of intelligence data generated by contemporary threat notification systems [6] (Figure 2 presents the details).

The main steps of text mining technology applied to threat intelligence notification are as follows: after selecting the text data, the text is first processed by tokenization [7], followed by lemmatization and stop-word removal, and then converted into a representation that the classifiers can distinguish by giving appropriate weights to the different word frequency computation methods. Then, the model of the classifiers can be trained before finally presenting the text mining results [8].

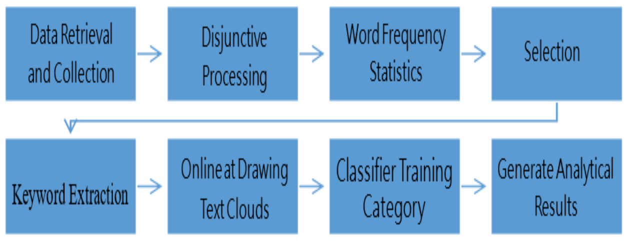

Data Retrieval and Collection
Disjunctive Processing
Word Frequency Statistics
Selection
Keyword Extraction
Online at Drawing Text Clouds
Classifier Training Category
Generate Analytical Results

Figure 2. Text mining flowchart (Source: Hyeon-Yi Lin, 2018)

2.3 Integrating Text Mining and Machine Learning

The combined application of text mining and machine learning technologies for threat intelligence mining and security risk elimination has become a significant trend in information security [9]. Text mining is leveraged to analyze vast amounts of text data to identify potential threat intelligence and attack patterns, helping organizations timely identify threat trends and behavioral characteristics. On the other hand, machine learning utilizes big data and pattern recognition technologies to achieve automated detection and defense against intrusions and attacks, thereby mitigating security risks and associated losses. This section summarizes the literature highlighting the advantages of combining textual analysis techniques with machine learning, specifically detailing how this integration can improve detection accuracy, reduce the false alarm rate, and accelerate the speed of analysis [10].

Text processing and machine learning techniques have a wide range of applications in the field of information security and, can effectively improve the identification of and response to threats. Automated cyber security tools and techniques based on machine learning algorithms have successfully mitigated the severity of cyber threats and, helped to avoid incidents. By using training data, the time required for predicting and preventing network surveillance and threat detection is reduced. Attack prevention systems can be rapidly developed using various machine learning methods. An effective Intrusion Detection Strategy (IDS) must be capable of detecting diverse forms of attacks, particularly those incorporating integrated avoidance strategies and high-protection security countermeasures [11] (Figure 3 provides an illustration).

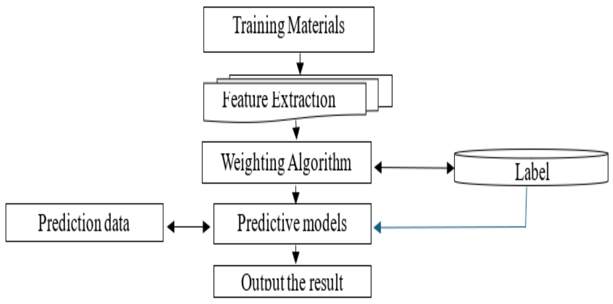


Figure 3. Text mining and machine learning (Source: Author’s own representation)

3 System Implementation and Case Validation

This section details the implementation of the proposed system and the design and operation of its constituent modules. The system architecture comprises a Data Collection and Preprocessing Module, a Feature Engineering Module, and a Machine Learning and Training Module. These components function in tandem to create the entire threat mapping and incident elimination system. The detailed process is illustrated in Figure 4 [10].

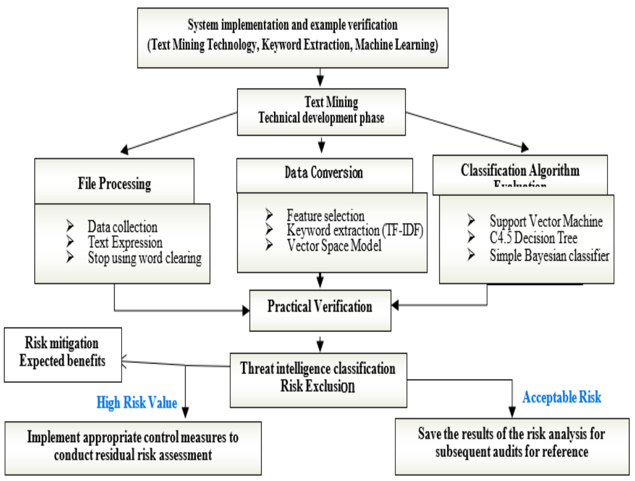

System implementation and example verification
(Text Mining Technology, Keyword Extraction, Machine Learning)
Text Mining
Technical development phase
File Processing
Data collection
Text Expression
Stop using word clearing
Data Conversion
Feature selection
Keyword extraction (TF-IDF)
Vector Space Model
Classification Algorithm
Support Vector Machine
C4.5 Decision Tree
Simple Bayesian classifier
Practical Verification
Risk mitigation
Expected benefits
Threat intelligence classification
Risk Exclusion
High Risk Value
Acceptable Risk
Implement appropriate control measures to conduct residual risk assessment
Save the results of the risk analysis for subsequent audits for reference

Figure 4. System architecture (Source: Author’s own representation)

(1) Data Preprocessing

The actual system data utilized in this study were obtained from the N-ISAC Threat Intelligence Reporting Data. The dataset encompasses 1,628 cases, categorized as intrusion attempts, intrusion attacks, and system vulnerabilities. Among them, 93.62% were intrusion attempts, 3.56% were intrusion attacks, and 2.46% were system vulnerabilities, totaling 1,628 cases. In order to improve the efficiency of machine learning, we first manually categorized the data into two categories: first-line maintenance cases and second-line maintenance quantities. The categorization criteria followed the threat information risk level methodology described in Section 3.4 of this study, using Grade B or above as the threshold for high-risk scoring. Following this examination, 1,228 cases were deemed below Grade B, and 400 cases were classified as Grade B or above. The experimental dataset was constructed using only the 400 high-risk cases (Grade B or above). Subsequently, the data were segmented based on criteria such as vulnerability severity, high attack source, attack complexity, privilege obtained, attack program release status, media/community disclosure status, patch unreleased status, and a list of successfully exploited vulnerabilities, resulting in the 1,628 data points summarized in Table 1 [12].

Table 1. Statistics of system operation data sources (Source: Author’s own representation)

Impact level / Type of event	Excluded Cases (below grade B)	Case Established V (Grade B or above)	Number of cases	Percentage %
Intrusion attempt V	1,170	354	1,524	93.62%
Intrusion Attack	31	27	58	3.56%
Malicious Content	1	0	1	0.06%
Service Interruption	2	0	2	0.12%
Content Security	2	0	2	0.12%
Fraudulent Attacks	1	0	1	0.06%
System Weaknesses	21	19	40	2.46%
Total	1,228	400	1,628	100%

(2) Textual Exploration and Feature Selection

In this study, we utilized the R programming language, which provides robust capabilities for statistical analysis, visualization, and data mining, to perform word segmentation, lexical annotation, and text preprocessing. Subsequently, the Term Frequency-Inverse Document Frequency (TF-IDF) method was employed for feature selection. Representative feature words—specifically, event subject/intelligence name, discovery time, content description/event description, impact level, destination IP/Port, and suggested measures/solutions—were finally selected via text mining to build the keyword corpus (Table 2) [13].

Considering the necessary accuracy for the machine learning model’s binary classification labeling, the experimental focus was placed on intrusion attempt cases. A case accepted by maintenance personnel was labeled ‘1’; otherwise, it was labeled ‘0’. In the dataset, 400 cases were labeled as ‘confirmed’ (i.e., requiring maintenance) and 1,228 as ‘excluded’. For the confirmed cases (Category 1 data), each case had at least one paragraph labeled ‘1’, indicating that maintenance personnel identified at least one critical feature value per case. The details are presented in Table 1 and Table 3.

Table 2. Characteristics of threat expert model categories (Source: Author’s own representation)

Item	Type	Categorical Feature	Definitions
1	Attack surface	Vulnerability severity:High	The possible impact severity of the vulnerability
2		Attack source: Internet	The source from which the attacker initiated the attack
3		Attack complexity:Low	Difficulty of the technique required for the attack
4	Privilege Upgrade	Host or network equipment for the service	Target systems affected by the attack
5		No authorization required	Whether or not you need to be authorized to launch an attack
6		Remote arbitrary stroke codes	Attackers can execute arbitrary code remotely
7	Potential Utilization	Attack program has been released	Publicly available attack code or tools
8		Media/community disclosure	Whether the vulnerability has been publicized by the media or the community
9		Patch not released	Official patch for the vulnerability
10		List of known successfully exploited vulnerabilities	Included in the list of vulnerabilities known to have been successfully exploited

Table 3. Machine learning data tagging statistics (Source: Author’s own representation)

Segmentation	Intrusion attempts	First-Line Maintenance		Second-Line Maintenance		Mixed Data	
		acceptance	rule out	acceptance	rule out	acceptance	rule out
Attack Level	Vulnerability severity: High	300	54	280	74	580	128
	Attack source: Internet	289	65	269	85	558	150
	Attack complexity: Low	136	218	116	238	252	456
Privilege Upgrade	Host or network device	134	250	124	230	258	480
	No authorization required	26	328	40	314	66	642
	Remote arbitrary stroke codes	270	84	220	134	490	218
Potential Utilization	Attack program has been released	16	338	46	308	62	646
	Media/community disclosure	0	354	0	354	0	708
	Patch not released	268	86	238	116	506	202
	Successful exploitation of vulnerability list	102	252	142	212	244	464
Total		1541	2029	1475	2065	3016	4094

(3) Classification method modeling

In this study, the first-line and second-line maintenance data were merged after using CkipTagger (a Traditional Chinese word segmentation tool) for tokenization and stop word exclusion. Given that information security personnel often reference similar past incidents in maintenance practice, the dataset was organized chronologically based on the case establishment time. The files were then automatically split into a two-thirds training set and a one-third testing set using the Support Vector Machine (SVM) classifier. Subsequently, TF-IDF was applied to quantify the words, from which 5,096 feature vectors were extracted. The top 30 feature words were statistically analyzed based on Term Frequency (TF), Inverse Document Frequency (IDF), and the product of the two (TF IDF), respectively. These top 30 TF-IDF feature terms included: terms describing the attack surface (vulnerability, source of the attack, complexity of the attack, social engineering, and ransomware, and privilege escalation); terms related to privilege (jailbreak, low privilege, and source code); and terms related to potential exploits (attacker, media, community, and vulnerability lists), among others [14].

(4) System implementation

In this study, the established threat information mining and security risk elimination system was successfully implemented. The system integrates core functional modules, including data import, text mining, feature selection, and classification method modeling, all accessible via a user-friendly interface.

(5) Case validation

The purpose of this step was to evaluate the correctness, accuracy, and recall of the different classification methods, specifically to identify the methods with the optimal detection accuracy and performance for malicious website detection. In the case validation, real threat notifications were used to evaluate the system’s performance and accuracy. By comparing the classification results with the actual outcomes, the system’s capabilities in threat information exploration and security risk elimination were validated. This system implementation and case validation process confirmed the validity and feasibility of the research methodology, providing an in-depth understanding of its practical application value in threat intelligence mapping and information security risk elimination.

3.1 System Analysis and Programming

This phase covers the system analysis and programming, primarily consisting of the data collection and preprocessing module, feature engineering, and the machine learning and training module. A complete threat mapping and security risk elimination system is constructed herein.

3.1.1 Data Collection and Preprocessing Module

The Data Collection and Preprocessing Module constitutes the foundation of the entire system. Its primary task is to collect data from diverse sources and preprocess them to ensure data quality and consistency. As shown in Figure 5, this module is systematically divided into three sub-modules: data collection, data preprocessing, and feature engineering.

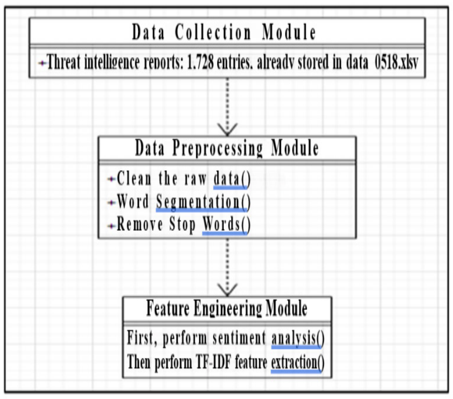

Data Collection Module
+Threat intelligence reports: 1,728 entries, already stored in data 0518.xlsx
Data Preprocessing Module
+Clean the raw data()
+Word Segmentation()
+Remove Stop Words()
Feature Engineering Module
First, perform sentiment analysis()
Then perform TF-IDF feature extraction()

Figure 5. Data collection and preprocessing workflow (Source: Author’s own representation)

(1) Data Collection Module

Data collection and preprocessing are indispensable steps in threat mapping research. Initially, a working directory was set up, and the raw Excel data were read using the readxl suite in R. To process the text data more efficiently, several R packages, including sentimentr, doParallel, foreach, and jiebaR, were introduced.

In the data collection module, the threat email data file was first loaded. All text fields were converted to lowercase, and punctuation marks were removed to ensure data consistency and accuracy in subsequent analysis. A self-defined sentiment lexicon, based on the maintenance staff's past experience, was also loaded to effectively identify key sentiment indicators. For Chinese text processing, the jiebaR suite was used to perform lexical segmentation, converting the text into a collection of analysable words. Upon completion of word segmentation, the occurrence frequency of all words was calculated to serve as the basis for subsequent sentiment analysis. Table 4 provides further details.

Table 4. Word frequency statistics (Source: Author's own representation)

Rank	Word	Frequency	Rank	Word	Frequency	Rank	Word	Frequency	Rank	Word	Frequency
1	Intelligence	623	21	Alcohol	58	41	Cause	29	61	Reduce	10
2	Main Cause	375	22	Implant	56	42	Property	28	62	Abnormal	9
3	Institution	267	23	Tiger	54	43	Loss	27	63	Slow	9
4	External	245	24	Program	52	44	Game	26	64	Location	9
5	Discover	217	25	Possibly	50	45	Social	25	65	Listed	9
6	Expensive	209	26	As	48	46	Website	24	66	Blacklist	9
7	Company	198	27	Relay Station	47	47	Pass	23	67	Up	8
8	Corporation	187	28	Will	46	48	Steal	22	68	Part	8
9	Limited Company	157	29	Below	45	49	Internet	21	69	Website	8
10	User	148	30	Risk	44	50	Attack	20	70	Access	8
11	Information	95	31	Personal Data	42	51	Exposure	19	71	Site	8
12	Device	90	32	And	40	52	Special	18	72	Account	8
13	Suspected	85	33	Privacy	38	53	Document	17	73	Personal Privacy	8
14	Infection	80	34	Leak	36	54	Confidential	16	74	Eliminate	7
15	Become	75	35	Path	35	55	Photo	15	75	Computer	7
16	Internet	70	36	Steal	34	56	Suffer	14	76	Computation	7
17	Member	68	37	Sell	32	57	Computer	13	77	Speed	7
18	External Communication	65	38	Financial Transaction	31	58	Resource	12	78	Become	7
19	Conduct	63	39	Data	30	59	Path	11	79	Internet	7
20	Attack	60	40	Information	30	60	Steal	11	80	Attack	7

(2) Data Preprocessing Module

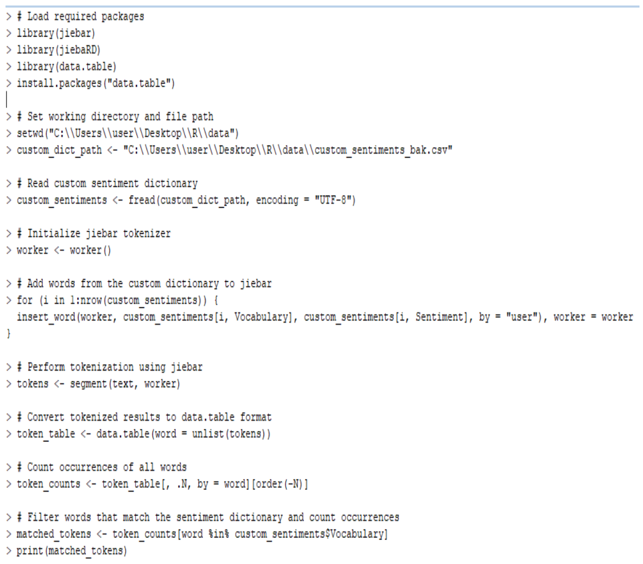
The main purpose of the data preprocessing module is to process the text data that have been divided into words, filter out the words that match the sentiment lexicon, and count them to ensure that high quality data are used in the subsequent sentiment analysis. In order to more accurately reflect the sentiment tendency of the text, sentiment calculation is used to calculate the sentiment value of each word. Data cleaning, missing value filling, and data standardization are performed during the process. A customized sentiment dictionary is used to process the sentiment information in the data, which includes sentiment words and their corresponding sentiment values set according to the needs of a specific domain, which helps the sentiment trends in the data to be more accurately analyzed and understood (see Table 5).

Table 5. Self-defined sentiment lexicon (Source: Author's own representation)

Item	Vocabulary	Emotion	interpretation	Item	Vocabulary	Emotion	Interpretation
1	Source of the attack	Negative	Indicates that there is malicious intent	16	Service interruptions	Negative	The system is not working properly
2	Network vulnerabilities	Negative	Weaknesses in the system or application	17	Denial-of-Service attack	Negative	Make the system unserviceable
3	Severity (High, Urgent, Critical)	Negative	Indicates the impact of the vulnerability	18	A patch has been released	Positive	The vulnerability has been fixed
4	Attack complexity: Low	Negative	Attackers are easy to exploit	19	Security updates	Positive	Improve system security
5	The host or network device of the Service	Negative	Attack the target	20	Bug bounty scheme	Positive	Encourage safety research
6	No permission is required	Negative	The attacker does not require special privileges	21	Security review	Positive	Identify potential vulnerabilities
7	Execute code remotely and arbitrarily	Negative	The attacker can control the system remotely	22	Best practices	Positive	Safety guidelines
8	The attacker has been released	Negative	Attack tools are publicly available	23	Multi-factor authentication	Positive	Enhance account security
9	The media/community has revealed	Negative	Vulnerability information is disclosed, increasing the risk of attacks	24	Intrusion detection system	Positive	Monitor for suspicious activity
10	The patch has not been released	Negative	There is no solution available	25	Security information and event management	Positive	Centrally manage security incidents
11	List of known successful exploits	Negative	Attack tactics are known to exist	26	Intimidation information	Positive	Stay up-to-date on the latest threats
12	Zero-day attacks	Negative	Unknown vulnerability, extremely high risk	27	Security awareness training	Positive	Increase employee safety awareness
13	Exploits	Negative	Exploit vulnerabilities to carry out attacks	28	Vulnerability scanning	Positive	Check system for vulnerabilities on a regular basis
14	Evil software	Negative	Malicious code	29	Penetration testing	Positive	Simulate attacks to test defenses
15	Data leakage	Negative	Confidential information is stolen				

(3) Feature Engineering Module

The main objective of the Feature Engineering Module is to extract useful features from the raw data for subsequent machine learning model training. This process involves feature selection, feature transformation, and feature construction. Feature selection aims to remove redundant or irrelevant features, retaining only the most valuable ones for model prediction. Feature transformation involves converting original features into new representations (e.g., via logarithmic transformation or standardization) to improve feature usability. Feature construction enriches the feature set by generating new features, such as interactive or aggregated features (see Figure 6 for details).



```
> # Load required packages
> library(jiebar)
> library(jiebaRD)
> library(data.table)
> install.packages("data.table")
|
> # Set working directory and file path
> setwd("C:\\Users\\user\\Desktop\\R\\data")
> custom_dict_path <- "C:\\Users\\user\\Desktop\\R\\data\\custom_sentiments_bak.csv"

> # Read custom sentiment dictionary
> custom_sentiments <- fread(custom_dict_path, encoding = "UTF-8")

> # Initialize jiebar tokenizer
> worker <- worker()

> # Add words from the custom dictionary to jiebar
> for (i in 1:nrow(custom_sentiments)) {
  insert_word(worker, custom_sentiments[i, Vocabulary], custom_sentiments[i, Sentiment], by = "user"), worker = worker
}

> # Perform tokenization using jiebar
> tokens <- segment(text, worker)

> # Convert tokenized results to data.table format
> token_table <- data.table(word = unlist(tokens))

> # Count occurrences of all words
> token_counts <- token_table[, .N, by = word][order(-N)]

> # Filter words that match the sentiment dictionary and count occurrences
> matched_tokens <- token_counts[word %in% custom_sentiments$Vocabulary]
> print(matched_tokens)
```

Figure 6. Data preprocessing feature engineering programming language (Source: Author's own representation)

3.1.2 Machine Learning and Training Modules

This section addresses the system analysis and programming of the Machine Learning and Training Module. Figure 7 illustrates the operational flowchart, which covers three main components: model training, model optimization, and model evaluation.

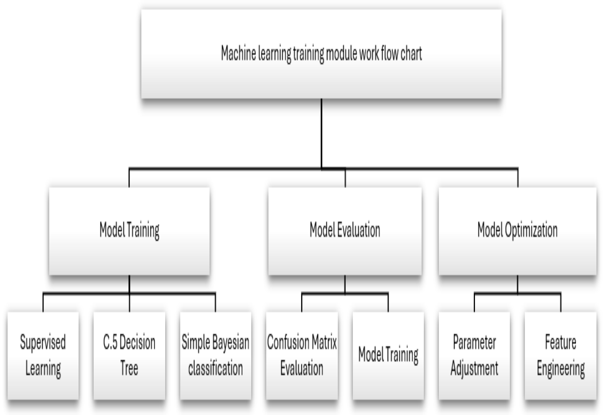
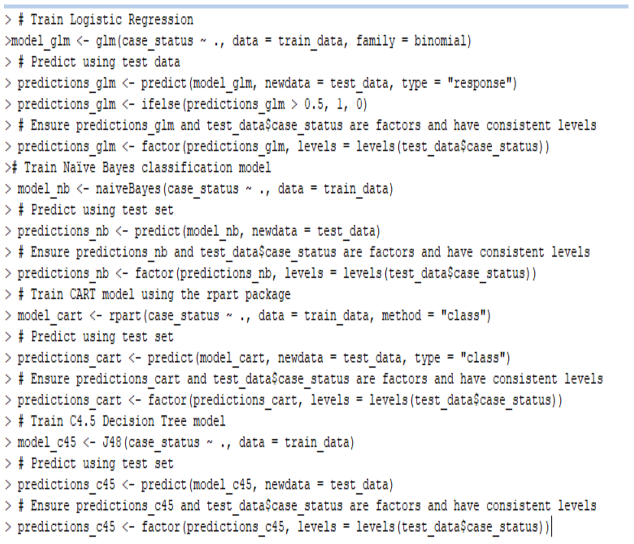


Figure 7. Machine learning and training module architecture (Source: Author’s own representation)

(1) Model Training

I In the model training stage, a supervised learning approach is used to build the prediction model, utilizing algorithms such as the C4.5 decision tree, CART decision tree, and logistic regression. These algorithms can handle various data types and provide accurate prediction results. Figure 8 illustrates the programming language execution process.

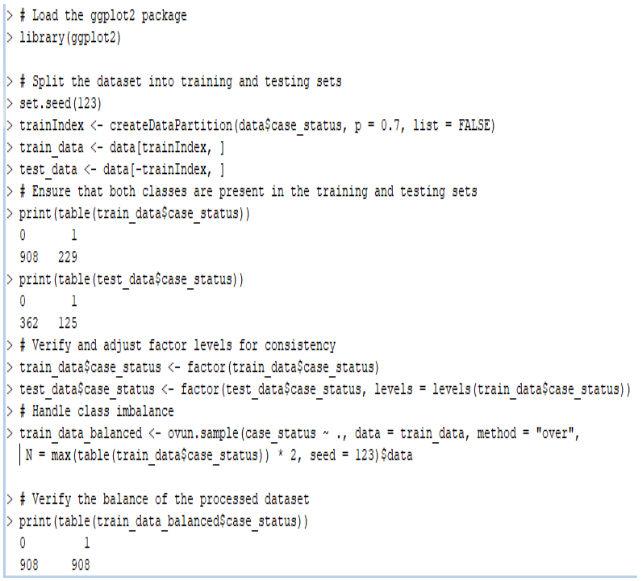


```
> # Train Logistic Regression
>model_glm <- glm(case_status ~ ., data = train_data, family = binomial)
> # Predict using test data
> predictions_glm <- predict(model_glm, newdata = test_data, type = "response")
> predictions_glm <- ifelse(predictions_glm > 0.5, 1, 0)
> # Ensure predictions_glm and test_data$case_status are factors and have consistent levels
> predictions_glm <- factor(predictions_glm, levels = levels(test_data$case_status))
># Train Naïve Bayes classification model
> model_nb <- naiveBayes(case_status ~ ., data = train_data)
> # Predict using test set
> predictions_nb <- predict(model_nb, newdata = test_data)
> # Ensure predictions_nb and test_data$case_status are factors and have consistent levels
> predictions_nb <- factor(predictions_nb, levels = levels(test_data$case_status))
> # Train CART model using the rpart package
> model_cart <- rpart(case_status ~ ., data = train_data, method = "class")
> # Predict using test set
> predictions_cart <- predict(model_cart, newdata = test_data, type = "class")
> # Ensure predictions_cart and test_data$case_status are factors and have consistent levels
> predictions_cart <- factor(predictions_cart, levels = levels(test_data$case_status))
> # Train C4.5 Decision Tree model
> model_c45 <- J48(case_status ~ ., data = train_data)
> # Predict using test set
> predictions_c45 <- predict(model_c45, newdata = test_data)
> # Ensure predictions_c45 and test_data$case_status are factors and have consistent levels
> predictions_c45 <- factor(predictions_c45, levels = levels(test_data$case_status))
```

Figure 8. Machine learning training module program (Source: Author’s own representation)

(2) Model Optimization

Model optimization is the process of improving model performance through parameter tuning and refined feature engineering. Cross-validation techniques are used to ensure the model’s generalization ability and prevent issues like overfitting or underfitting. Grid Search or Random Search is employed to identify the optimal combination of hyperparameters, and Feature Engineering is leveraged to select and construct features that maximize model performance (see Figure 9 for details).

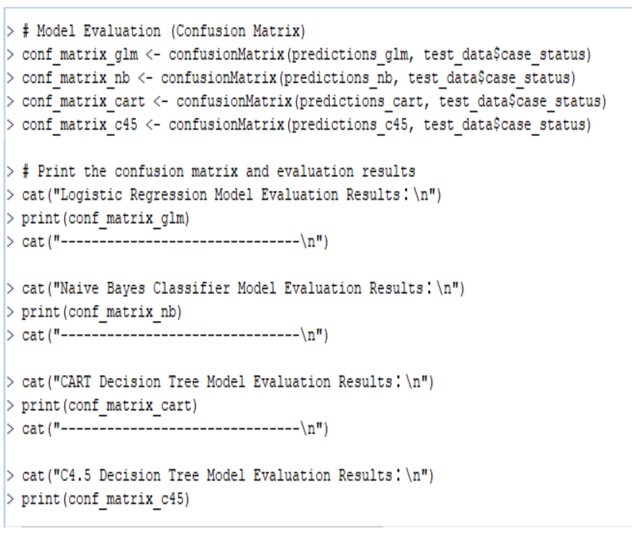


```
> # Load the ggplot2 package
> library(ggplot2)

> # Split the dataset into training and testing sets
> set.seed(123)
> trainIndex <- createDataPartition(data$case_status, p = 0.7, list = FALSE)
> train_data <- data[trainIndex, ]
> test_data <- data[-trainIndex, ]
> # Ensure that both classes are present in the training and testing sets
> print(table(train_data$case_status))
  0      1
  908   229
> print(table(test_data$case_status))
  0      1
  362   125
> # Verify and adjust factor levels for consistency
> train_data$case_status <- factor(train_data$case_status)
> test_data$case_status <- factor(test_data$case_status, levels = levels(train_data$case_status))
> # Handle class imbalance
> train_data_balanced <- ovun.sample(case_status ~ ., data = train_data, method = "over",
  N = max(table(train_data$case_status)) * 2, seed = 123)$data

> # Verify the balance of the processed dataset
> print(table(train_data_balanced$case_status))
  0       1
  908     908
```

Figure 9. Model optimization program (Source: Author’s own representation)

(3) Model Evaluation

The confusion matrix is used in the model evaluation step to assess the model’s predictive performance. The confusion matrix provides the counts of true positives, false positives, true negatives, and false negatives. These values are then used to calculate key evaluation metrics, including accuracy, recall (sensitivity), and F1 score, thereby comprehensively evaluating the model’s predictive capability (Figure 10).



```
> # Model Evaluation (Confusion Matrix)
> conf_matrix_glm <- confusionMatrix(predictions_glm, test_data$case_status)
> conf_matrix_nb <- confusionMatrix(predictions_nb, test_data$case_status)
> conf_matrix_cart <- confusionMatrix(predictions_cart, test_data$case_status)
> conf_matrix_c45 <- confusionMatrix(predictions_c45, test_data$case_status)

> # Print the confusion matrix and evaluation results
> cat("Logistic Regression Model Evaluation Results:\n")
> print(conf_matrix_glm)
> cat("------------------------------\n")

> cat("Naive Bayes Classifier Model Evaluation Results:\n")
> print(conf_matrix_nb)
> cat("------------------------------\n")

> cat("CART Decision Tree Model Evaluation Results:\n")
> print(conf_matrix_cart)
> cat("------------------------------\n")

> cat("C4.5 Decision Tree Model Evaluation Results:\n")
> print(conf_matrix_c45)
```

Figure 10. Model evaluation program (Source: Author’s own representation)

3.2 Experimental Results and Predictive Analysis

This section presents a detailed performance analysis of several common forecasting models: the C4.5 Decision

Tree, Naïve Bayes, Logistic Regression, and CART Decision Tree models. The forecasting program workflow and the corresponding data tables for each model are illustrated in Figure 11 through Figure 21.

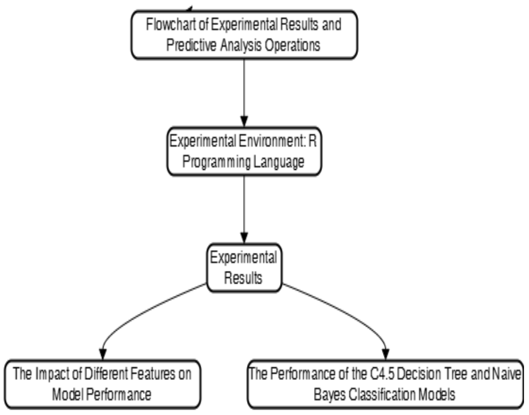

Flowchart of Experimental Results and Predictive Analysis Operations
Experimental Environment: R Programming Language
Experimental Results
The Impact of Different Features on Model Performance
The Performance of the C4.5 Decision Tree and Naive Bayes Classification Models

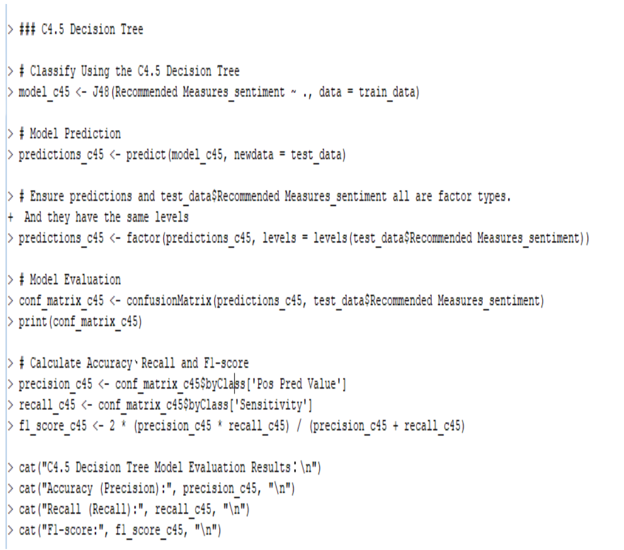
Figure 11. Experimental results and forecast analysis (Source: Author's own representation)

3.2.1 C4.5 Decision Tree Model

The C4.5 decision tree model was employed to categorize threat information, specifically to determine whether a case was established. A key strength of this model is its capability to effectively handle both discrete and continuous variables, as well as its robustness against missing values (as shown in Figure 12).

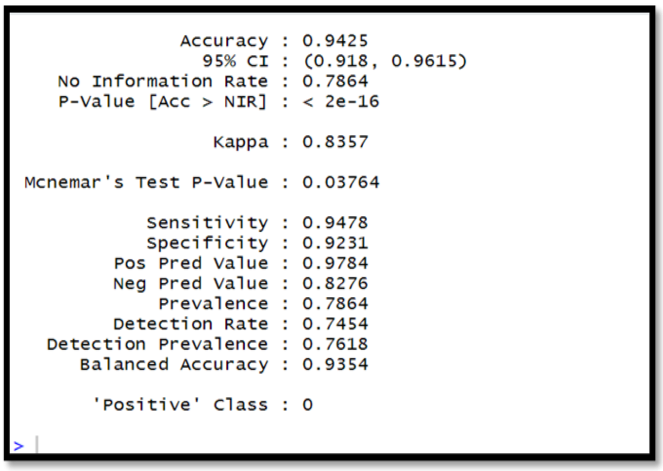
The experimental results demonstrate that the C4.5 decision tree model achieved an accuracy rate of 94.25% in threat information exploration. Its performance metrics include a sensitivity of 94.78%, a specificity of 92.31%, and a Kappa value of 0.8357. The model shows significant advantages in balancing prediction accuracy with interpretability, establishing it as a reliable and practical tool for threat classification. This provides crucial support for efficient threat management and response strategies (see Figure 13).

Model Training and Prediction Process



```
> ### C4.5 Decision Tree

> # Classify Using the C4.5 Decision Tree
> model_c45 <- J48(Recommended Measures_sentiment ~ ., data = train_data)

> # Model Prediction
> predictions_c45 <- predict(model_c45, newdata = test_data)

> # Ensure predictions and test_data$Recommended Measures_sentiment all are factor types.
+  And they have the same levels
> predictions_c45 <- factor(predictions_c45, levels = levels(test_data$Recommended Measures_sentiment))

> # Model Evaluation
> conf_matrix_c45 <- confusionMatrix(predictions_c45, test_data$Recommended Measures_sentiment)
> print(conf_matrix_c45)

> # Calculate Accuracy、Recall and F1-score
> precision_c45 <- conf_matrix_c45$byClass['Pos Pred Value']
> recall_c45 <- conf_matrix_c45$byClass['Sensitivity']
> f1_score_c45 <- 2 * (precision_c45 * recall_c45) / (precision_c45 + recall_c45)

> cat("C4.5 Decision Tree Model Evaluation Results：\n")
> cat("Accuracy (Precision):", precision_c45, "\n")
> cat("Recall (Recall):", recall_c45, "\n")
> cat("F1-score:", f1_score_c45, "\n")
```

Figure 12. C4.5 decision tree classification model prediction program**:** model training and prediction process (Source: Authors' own representation)



```
               Accuracy : 0.9425
                 95% CI : (0.918, 0.9615)
    No Information Rate : 0.7864
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8357

 Mcnemar's Test P-Value : 0.03764

            Sensitivity : 0.9478
            Specificity : 0.9231
         Pos Pred Value : 0.9784
         Neg Pred Value : 0.8276
             Prevalence : 0.7864
         Detection Rate : 0.7454
   Detection Prevalence : 0.7618
      Balanced Accuracy : 0.9354

       'Positive' Class : 0

>
```

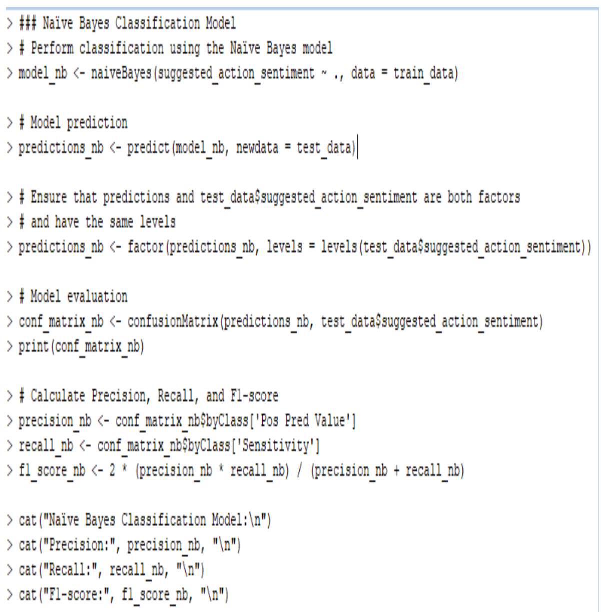
Figure 13. C4.5 decision tree confusion matrix evaluation results (Source: Author's own representation)

3.2.2 Naïve Bayes Model

The Naïve Bayes model's rapid prediction capability in threat mapping stems from its simplifying assumptions, efficient computational process, and efficacy in handling high-dimensional data. These attributes make it particularly well-suited for real-time threat detection, initial screening, and high-frequency prediction applications. Despite its inherent assumptions, the model provides accurate classification results in practical applications, making it a highly appropriate algorithm for threat mapping (Figure 14).

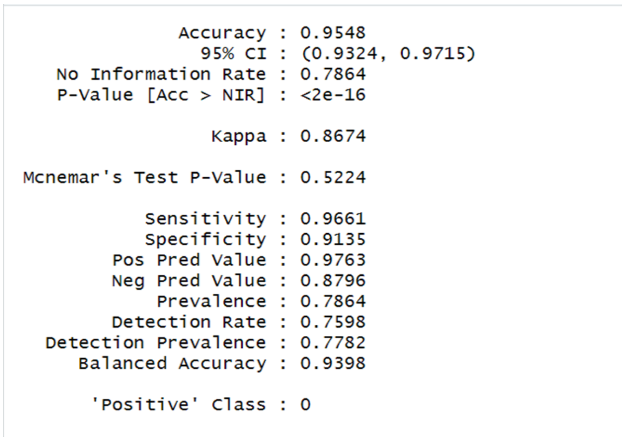
The experimental results indicate that the Naïve Bayes model's accuracy in threat mapping reaches 95.48%. Its performance metrics are a sensitivity of 96.61%, a specificity of 91.35%, and a Kappa value of 0.8674. This demonstrates a significant advantage in balancing accuracy and speed, confirming its role as a reliable and practical tool for threat classification (Figure 15).

Model Training and Prediction Process



```
> ### Naïve Bayes Classification Model
> # Perform classification using the Naïve Bayes model
> model_nb <- naiveBayes(suggested_action_sentiment ~ ., data = train_data)

> # Model prediction
> predictions_nb <- predict(model_nb, newdata = test_data)

> # Ensure that predictions and test_data$suggested_action_sentiment are both factors
> # and have the same levels
> predictions_nb <- factor(predictions_nb, levels = levels(test_data$suggested_action_sentiment))

> # Model evaluation
> conf_matrix_nb <- confusionMatrix(predictions_nb, test_data$suggested_action_sentiment)
> print(conf_matrix_nb)

> # Calculate Precision, Recall, and F1-score
> precision_nb <- conf_matrix_nb$byClass['Pos Pred Value']
> recall_nb <- conf_matrix_nb$byClass['Sensitivity']
> f1_score_nb <- 2 * (precision_nb * recall_nb) / (precision_nb + recall_nb)

> cat("Naïve Bayes Classification Model:\n")
> cat("Precision:", precision_nb, "\n")
> cat("Recall:", recall_nb, "\n")
> cat("F1-score:", f1_score_nb, "\n")
```

Figure 14. Naïve Bayes classification model forecasting program: model training and prediction process (Source: Compiled by the authors)



```
               Accuracy : 0.9548
                 95% CI : (0.9324, 0.9715)
    No Information Rate : 0.7864
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8674

 Mcnemar's Test P-Value : 0.5224

            Sensitivity : 0.9661
            Specificity : 0.9135
         Pos Pred Value : 0.9763
         Neg Pred Value : 0.8796
             Prevalence : 0.7864
         Detection Rate : 0.7598
   Detection Prevalence : 0.7782
      Balanced Accuracy : 0.9398

       'Positive' Class : 0
```

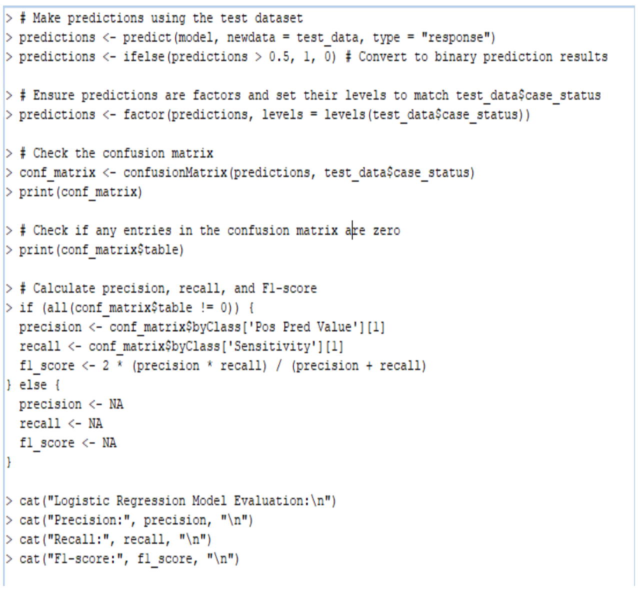
Figure 15. Results of confusion matrix evaluation for the Naïve Bayes classification model (Source: Author's own representation)

3.2.3 Logistic Regression Model

In this study, the Logistic Regression model was utilized for threat intelligence exploration, primarily for threat event classification and feature selection. By learning the relationship between features and threat events, the model effectively categorizes threat information and offers explainability regarding the impact of important features, aiding security experts in data analysis (see Figure 16).

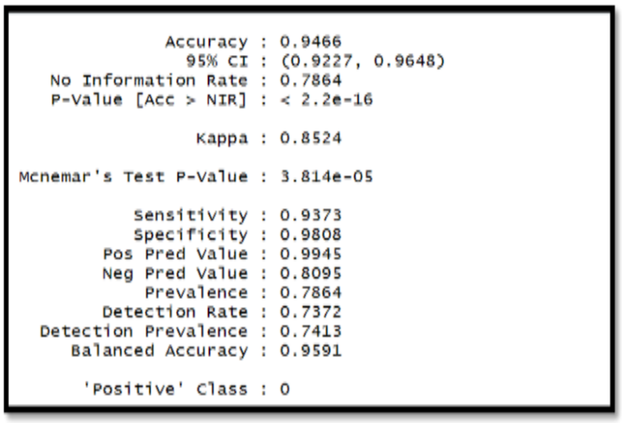
The experimental results show that the Logistic Regression model achieved an accuracy of 94.66% in threat mapping. Its performance metrics are a sensitivity of 93.73%, a specificity of 98.08%, and a Kappa value of 0.8524. The high specificity highlights its ability to correctly identify non-threat cases. This model thus offers significant advantages in balancing accuracy and interpretability, making it a reliable and practical tool for threat classification and providing essential support for efficient threat management and response strategies (Figure 17).

Model Training and Prediction Process



```
> # Make predictions using the test dataset
> predictions <- predict(model, newdata = test_data, type = "response")
> predictions <- ifelse(predictions > 0.5, 1, 0) # Convert to binary prediction results

> # Ensure predictions are factors and set their levels to match test_data$case_status
> predictions <- factor(predictions, levels = levels(test_data$case_status))

> # Check the confusion matrix
> conf_matrix <- confusionMatrix(predictions, test_data$case_status)
> print(conf_matrix)

> # Check if any entries in the confusion matrix are zero
> print(conf_matrix$table)

> # Calculate precision, recall, and F1-score
> if (all(conf_matrix$table != 0)) {
  precision <- conf_matrix$byClass['Pos Pred Value'][1]
  recall <- conf_matrix$byClass['Sensitivity'][1]
  f1_score <- 2 * (precision * recall) / (precision + recall)
} else {
  precision <- NA
  recall <- NA
  f1_score <- NA
}

> cat("Logistic Regression Model Evaluation:\n")
> cat("Precision:", precision, "\n")
> cat("Recall:", recall, "\n")
> cat("F1-score:", f1_score, "\n")
```

Figure 16. Logistic regression model prediction program: model training and prediction process (Source: Author's own representation)



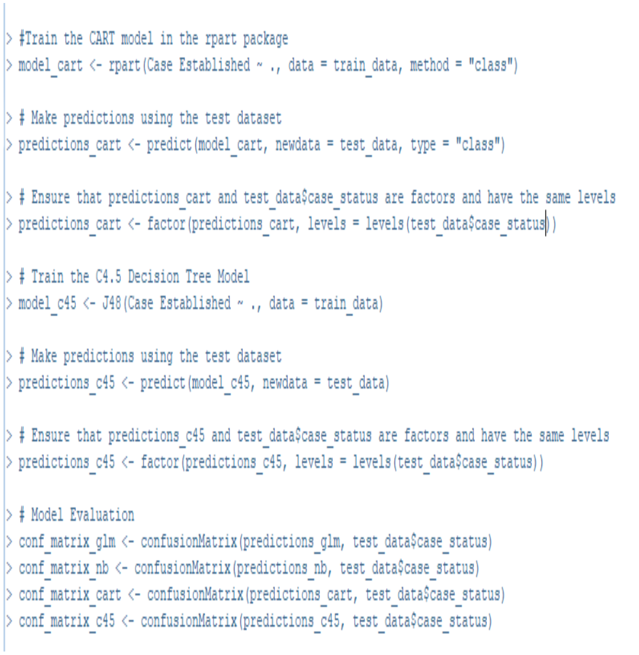
```
               Accuracy : 0.9466
                 95% CI : (0.9227, 0.9648)
    No Information Rate : 0.7864
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8524

 Mcnemar's Test P-Value : 3.814e-05

            Sensitivity : 0.9373
            Specificity : 0.9808
         Pos Pred Value : 0.9945
         Neg Pred Value : 0.8095
             Prevalence : 0.7864
         Detection Rate : 0.7372
   Detection Prevalence : 0.7413
      Balanced Accuracy : 0.9591

       'Positive' Class : 0
```

Figure 17. Results of confusion matrix evaluation for the logistic regression classification model (Source: Author's own representation)

3.2.4 CART Decision Tree Model

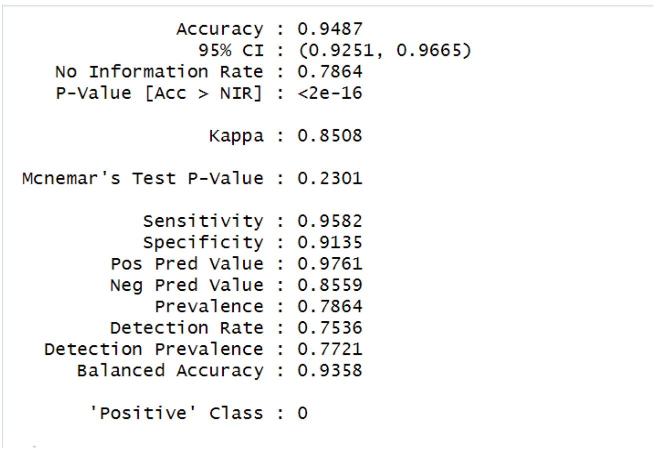
The CART decision tree model was effectively applied in this project to support threat information management and decision-making. Its inherent clear decision rules and efficient prediction capability enhance its application value and improve the efficiency and accuracy of threat mapping (Figure 18).

The experimental results show that the CART decision tree model has an accuracy of 94.87%, a sensitivity of 95.82%, a specificity of 91.35%, and a Kappa value of 0.8508. These metrics indicate excellent accuracy and stability in categorizing threat information and provide reliable prediction results. The model plays a vital role in identifying potential threats and analyzing data characteristics, offering strong support for threat management. Its explicit decision rules and interpretability further enhance its application value (see Figure 19).



```
> #Train the CART model in the rpart package
> model_cart <- rpart(Case Established ~ ., data = train_data, method = "class")

> # Make predictions using the test dataset
> predictions_cart <- predict(model_cart, newdata = test_data, type = "class")

> # Ensure that predictions_cart and test_data$case_status are factors and have the same levels
> predictions_cart <- factor(predictions_cart, levels = levels(test_data$case_status))

> # Train the C4.5 Decision Tree Model
> model_c45 <- J48(Case Established ~ ., data = train_data)

> # Make predictions using the test dataset
> predictions_c45 <- predict(model_c45, newdata = test_data)

> # Ensure that predictions_c45 and test_data$case_status are factors and have the same levels
> predictions_c45 <- factor(predictions_c45, levels = levels(test_data$case_status))

> # Model Evaluation
> conf_matrix_glm <- confusionMatrix(predictions_glm, test_data$case_status)
> conf_matrix_nb <- confusionMatrix(predictions_nb, test_data$case_status)
> conf_matrix_cart <- confusionMatrix(predictions_cart, test_data$case_status)
> conf_matrix_c45 <- confusionMatrix(predictions_c45, test_data$case_status)
```

Figure 18. CART decision tree model prediction program (Source: Author's own representation)



```
               Accuracy : 0.9487
                 95% CI : (0.9251, 0.9665)
    No Information Rate : 0.7864
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8508

 Mcnemar's Test P-Value : 0.2301

            Sensitivity : 0.9582
            Specificity : 0.9135
         Pos Pred Value : 0.9761
         Neg Pred Value : 0.8559
             Prevalence : 0.7864
         Detection Rate : 0.7536
   Detection Prevalence : 0.7721
      Balanced Accuracy : 0.9358

       'Positive' Class : 0
```

Figure 19. CART decision tree model confusion matrix evaluation results. (Source: Author's own representation)

3.2.5 Comparative Analysis of the Four Models

This project demonstrated the strong performance of the C4.5 decision tree, Simple Bayesian, logistic regression, and CART decision tree models. The C4.5 model stood out with high accuracy (95.07%), excellent sensitivity (96.96%), and strong explanatory properties, enabling a clear presentation of the decision logic and reliable support for threat information classification. While the Naïve Bayes Model's accuracy is competitive (95.48%) and its sensitivity is high (96.61%), its efficient computational speed and adaptability to high-dimensional data make it an ideal choice for rapid prediction. The logistic regression model performed well, showing high sensitivity (97.79%) and accuracy (94.66%), making it suitable for scenarios demanding high interpretability and accurate predictions. The CART decision tree model also demonstrated good accuracy (94.87%), sensitivity (95.82%), and specificity (91.35%), effectively identifying potential threats through explicit decision rules; however, careful consideration must be given to its sensitivity to data quality and the risk of over-simulation (overfitting) (Table 6).

Table 6. Comparison of experimental results of four major models (Source: Author's own representation)

Model Category	Note description	Decision Tree Model	Logic is the same	Simple shellfish classification	CART model
Accuracy	Accuracy	0.9425	0.9466	0.9548	0.9487
95% CI	Confidence interval	0.918,0.9615	0.9227,0.9648	0.9324,0.9715	0.9251,0.9665
No Information Rate		0.7864	0.7864	0.7864	0.7864
p-Value Acc>NIR]		<2e-16	<2.2e-15	<2e-16	<2e-16
Kappa		0.8357	0.8524	0.8674	0.8508
Mcnemar's Test P-Value	Test p-value	0.03764	3.814r-05	0.5224	0.2301
Sensitivity	Sensitivity	0.9478	0.9373	0.9661	0.9582
Specificity	Specificity	0.9231	0.9808	0.9135	0.9135
Pos Pred Value	Positive predictive value	0.9784	0.9945	0.9763	0.9761
Neg Pred Value	Negative predictive value	0.8276	0.8095	0.8796	0.8559
Prevalence		0.7864	0.7864	0.7864	0.7864
Detection Rate		0.7454	0.7372	0.7598	0.7864
Detection Prevalence		0.7618	0.7413	0.7782	0.7721
Balanced Accuracy	Balanced Accuracy	0.9354	0.9591	0.9398	0.9358

As noted by Singh et al. (2020) in "Intrusion Detection System: A Comparative Study of Machine Learning-Based IDS," confusion matrices are widely used to evaluate binary classifiers, covering both correctly and incorrectly classified sample sizes [15]. Their study highlights the use of accuracy, precision, recall (sensitivity), and -score metrics but emphasizes the critical issue of false positives and omissions in intrusion detection systems, particularly their performance when handling unknown attacks.

Consequently, we selected specificity (True Negative Rate) and sensitivity (True Positive Rate) as the core evaluation metrics to analyze the models' performance characteristics and real-world effectiveness in greater depth. This approach not only facilitates a comprehensive comparison of different machine learning models in intrusion detection but also provides a robust justification for the ultimate selection of logistic regression (Figure 20).

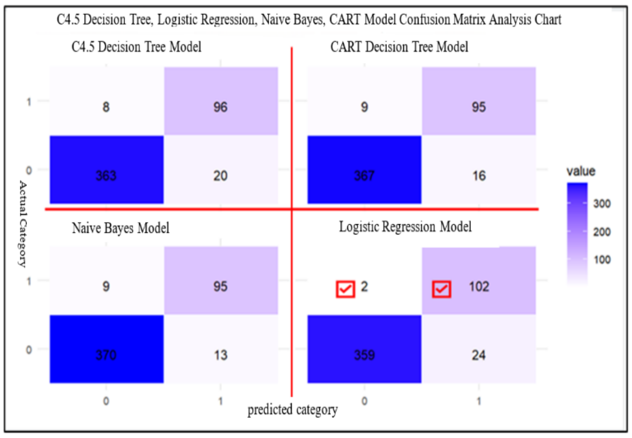


Figure 20. Results of confusion matrix evaluation of four major model predictions (Source: Author's own representation)

3.3 Case Studies (Three Cases)

In this study, a detailed practical validation of threat intelligence prospecting and information security risk elimination was conducted. To verify the effectiveness of the proposed method and model, three typical information security cases were selected: intrusion attempts, intrusion attacks, and malicious content (Figure 21).

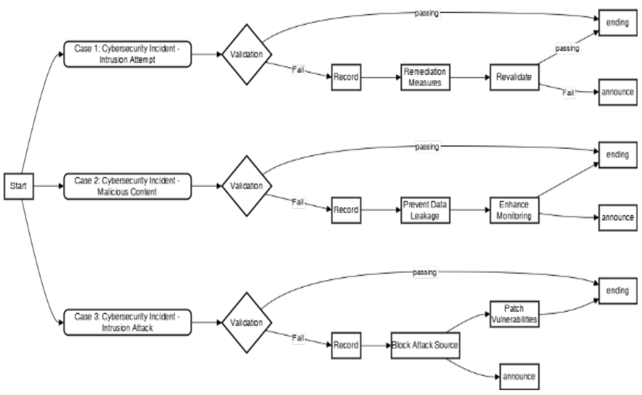
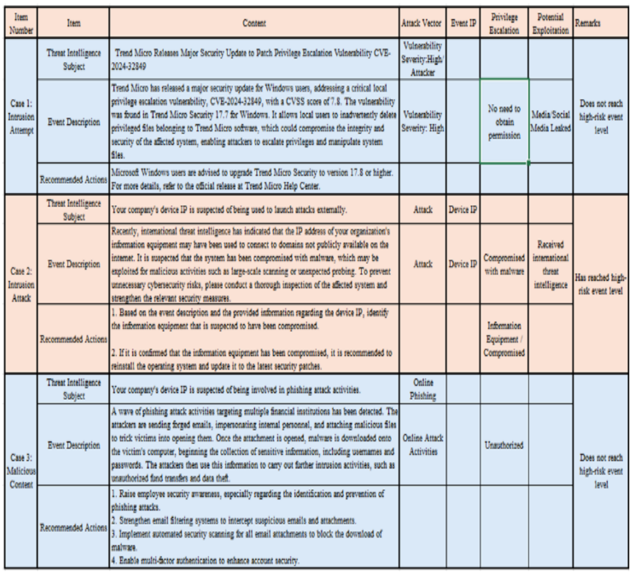


Figure 21. Case validation practice flowchart (Source: Author's own representation)

3.3.1 Case Study Analysis

To validate the effectiveness of the proposed model and method, this section presents an analysis of results from a case study. Applying the model to real-world scenarios allows us to observe its practical performance and further evaluate its feasibility and accuracy in threat detection and risk mitigation. The following content is structured into three parts: Section 3.3.1. provides a case study analysis of a representative intrusion incident, illustrating the model's practical application; Section 3.3.2. systematically compares the models' ability to identify potential threats through a threat risk assessment; and finally, Section 3.3.3. provides a code implementation example, demonstrating the model's operational process and application details to ensure the reproducibility and practical value of the research findings (Table 7) [10].

Table 7. Comparison of empirical analyses of the three case studies (Source: Author's own representation)



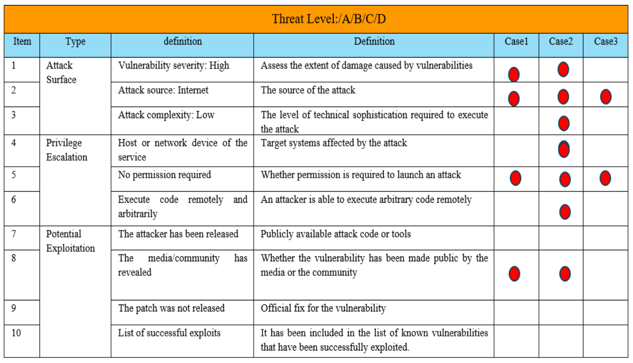
Item Number	Item	Content	Attack Vector	Event IP	Privilege Escalation	Potential Exploitation	Remarks
Case 1: Intrusion Attempt	Threat Intelligence Subject	Trend Micro Releases Major Security Update to Patch Privilege Escalation Vulnerability CVE-2024-32849	Vulnerability Severity:High/ Attacker				Does not reach high-risk event level
	Event Description	Trend Micro has released a major security update for Windows users, addressing a critical local privilege escalation vulnerability, CVE-2024-32849, with a CVSS score of 7.8. The vulnerability was found in Trend Micro Security 17.7 for Windows. It allows local users to inadvertently delete privileged files belonging to Trend Micro software, which could compromise the integrity and security of the affected system, enabling attackers to escalate privileges and manipulate system files.	Vulnerability Severity: High		No need to obtain permission	Media/Social Media Leaked	
	Recommended Actions	Microsoft Windows users are advised to upgrade Trend Micro Security to version 17.8 or higher. For more details, refer to the official release at Trend Micro Help Center.					
Case 2: Intrusion Attack	Threat Intelligence Subject	Your company's device IP is suspected of being used to launch attacks externally.	Attack	Device IP			Has reached high-risk event level
	Event Description	Recently, international threat intelligence has indicated that the IP address of your organization's information equipment may have been used to connect to domains not publicly available on the internet. It is suspected that the system has been compromised with malware, which may be exploited for malicious activities such as large-scale scanning or unexpected probing. To prevent unnecessary cybersecurity risks, please conduct a thorough inspection of the affected system and strengthen the relevant security measures.	Attack	Device IP	Compromised with malware	Received international threat intelligence	
	Recommended Actions	1. Based on the event description and the provided information regarding the device IP, identify the information equipment that is suspected to have been compromised. 2. If it is confirmed that the information equipment has been compromised, it is recommended to reinstall the operating system and update it to the latest security patches.			Information Equipment / Compromised		
Case 3: Malicious Content	Threat Intelligence Subject	Your company's device IP is suspected of being involved in phishing attack activities.	Online Phishing				Does not reach high-risk event level
	Event Description	A wave of phishing attack activities targeting multiple financial institutions has been detected. The attackers are sending forged emails, impersonating internal personnel, and attaching malicious files to trick victims into opening them. Once the attachment is opened, malware is downloaded onto the victim's computer, beginning the collection of sensitive information, including usernames and passwords. The attackers then use this information to carry out further intrusion activities, such as unauthorized fund transfers and data theft.	Online Attack Activities		Unauthorized		
	Recommended Actions	1. Raise employee security awareness, especially regarding the identification and prevention of phishing attacks. 2. Strengthen email filtering systems to intercept suspicious emails and attachments. 3. Implement automated security scanning for all email attachments to block the download of malware. 4. Enable multi-factor authentication to enhance account security.					

3.3.2 Threat Risk Level Assessment

In accordance with the NIST 2.0 framework, threat intelligence risks are categorized into four levels: A, B, C, and D. The corresponding response protocols and handling guidelines for each risk level are comprehensively outlined in Table 8.

- Case Study 1 was assessed as Level C.
- Case Study 2 was assessed as Level B.
- Case Study 3 was assessed as Level C.

Table 8. Assessment of threat risk levels for three cases (Source: Compiled by the author)

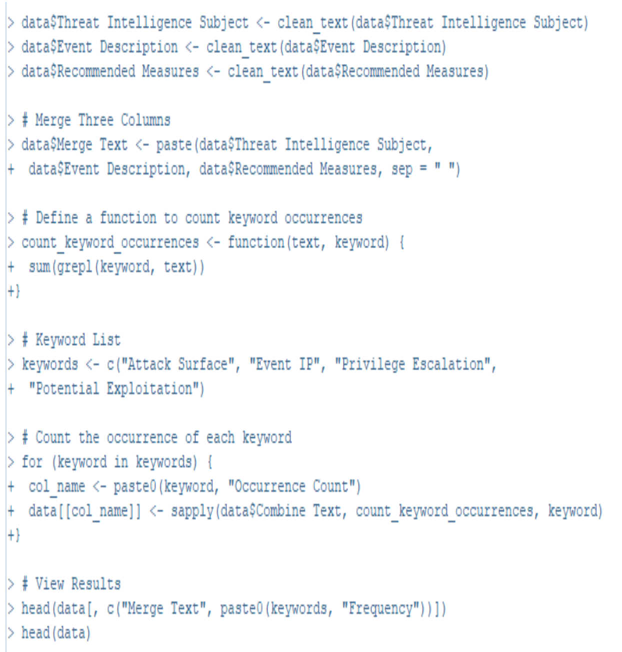


Threat Level:/A/B/C/D						
Item	Type	definition	Definition	Case1	Case2	Case3
1	Attack Surface	Vulnerability severity: High	Assess the extent of damage caused by vulnerabilities	●	●	
2		Attack source: Internet	The source of the attack	●	●	●
3		Attack complexity: Low	The level of technical sophistication required to execute the attack		●	
4	Privilege Escalation	Host or network device of the service	Target systems affected by the attack		●	
5		No permission required	Whether permission is required to launch an attack	●	●	●
6		Execute code remotely and arbitrarily	An attacker is able to execute arbitrary code remotely		●	
7	Potential Exploitation	The attacker has been released	Publicly available attack code or tools			
8		The media/community has revealed	Whether the vulnerability has been made public by the media or the community	●	●	
9		The patch was not released	Official fix for the vulnerability			
10		List of successful exploits	It has been included in the list of known vulnerabilities that have been successfully exploited.			

3.3.3 Program Code Implementation

T The following code illustrates the process for testing the intrusion attempt case using the logistic regression, Simple Bayesian, CART decision tree, and C4.5 decision tree models (Figure 22).

(1) Data Preprocessing

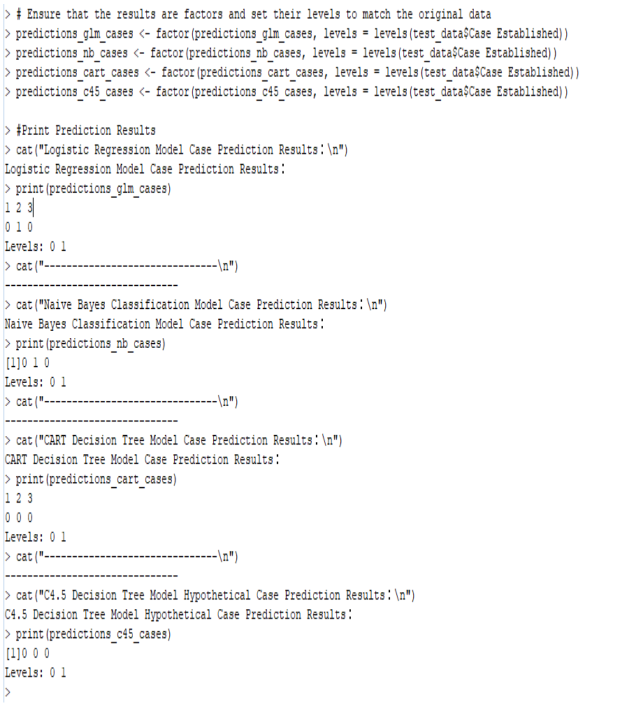


```
> data$Threat Intelligence Subject <- clean_text(data$Threat Intelligence Subject)
> data$Event Description <- clean_text(data$Event Description)
> data$Recommended Measures <- clean_text(data$Recommended Measures)

> # Merge Three Columns
> data$Merge Text <- paste(data$Threat Intelligence Subject,
+  data$Event Description, data$Recommended Measures, sep = " ")

> # Define a function to count keyword occurrences
> count_keyword_occurrences <- function(text, keyword) {
+  sum(grepl(keyword, text))
+}

> # Keyword List
> keywords <- c("Attack Surface", "Event IP", "Privilege Escalation",
+  "Potential Exploitation")

> # Count the occurrence of each keyword
> for (keyword in keywords) {
+  col_name <- paste0(keyword, "Occurrence Count")
+  data[[col_name]] <- sapply(data$Combine Text, count_keyword_occurrences, keyword)
+}

> # View Results
> head(data[, c("Merge Text", paste0(keywords, "Frequency"))])
> head(data)
```

Figure 22. Preprocessing program flow for validation of data

(2) Machine Learning Language (MLL) three-case validation system flow

The machine learning program flow for validating the three cases is shown in Figure 23.



```
> # Ensure that the results are factors and set their levels to match the original data
> predictions_glm_cases <- factor(predictions_glm_cases, levels = levels(test_data$Case Established))
> predictions_nb_cases <- factor(predictions_nb_cases, levels = levels(test_data$Case Established))
> predictions_cart_cases <- factor(predictions_cart_cases, levels = levels(test_data$Case Established))
> predictions_c45_cases <- factor(predictions_c45_cases, levels = levels(test_data$Case Established))

> #Print Prediction Results
> cat("Logistic Regression Model Case Prediction Results:\n")
Logistic Regression Model Case Prediction Results:
> print(predictions_glm_cases)
1 2 3
0 1 0
Levels: 0 1
> cat("------------------------------\n")
------------------------------
> cat("Naive Bayes Classification Model Case Prediction Results:\n")
Naive Bayes Classification Model Case Prediction Results:
> print(predictions_nb_cases)
[1]0 1 0
Levels: 0 1
> cat("------------------------------\n")
------------------------------
> cat("CART Decision Tree Model Case Prediction Results:\n")
CART Decision Tree Model Case Prediction Results:
> print(predictions_cart_cases)
1 2 3
0 0 0
Levels: 0 1
> cat("------------------------------\n")
------------------------------
> cat("C4.5 Decision Tree Model Hypothetical Case Prediction Results:\n")
C4.5 Decision Tree Model Hypothetical Case Prediction Results:
> print(predictions_c45_cases)
[1]0 0 0
Levels: 0 1
>
```

Figure 23. Machine learning program for validating three cases (Source: Author's own representation)

(3) Analysis of case verification test results

The analysis of the three case study validation test results is detailed in Figure 24.

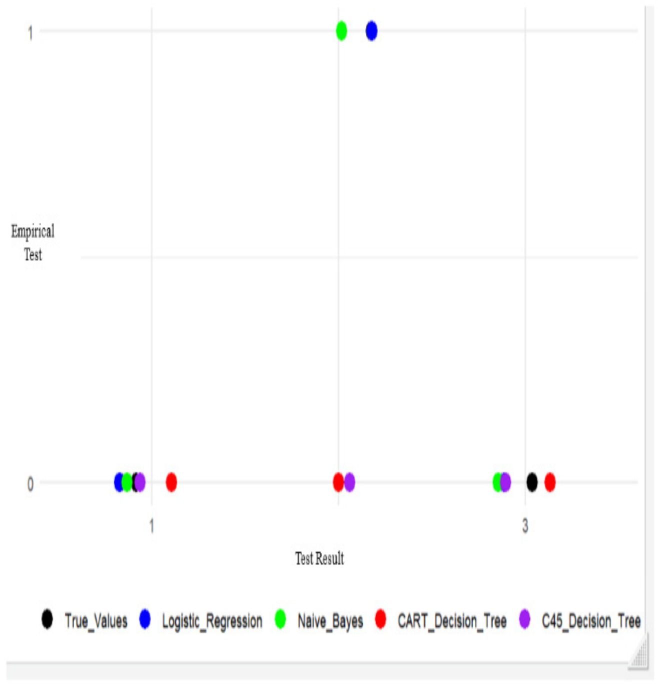


Figure 24. Analysis of three case study validation test results (Source: Compiled by the author)

4 Conclusion and Recommendations

4.1 Conclusion

This research presents an innovative integration of machine learning and text mining technologies for cybersecurity threat intelligence analysis. Utilizing the R programming language, we conducted in-depth and systematic data analysis of threat intelligence reports from competent authorities. This process involved extracting key features from massive textual information to rapidly identify potential security threats and system vulnerabilities. Our approach significantly reduces the time required for cybersecurity professionals to interpret and evaluate threat intelligence, providing accurate and actionable insights that assist in developing effective protective measures.

Furthermore, the model developed in this research incorporates automated threat detection capabilities, enabling real-time monitoring and analysis of new intelligence data to facilitate the early identification of potential attacks and mitigate the risk of cybersecurity incidents faced by enterprises. This innovative methodology not only improves the efficiency of threat detection and risk management but also provides cybersecurity professionals with a set of analytical tools directly applicable to their operational workflow. The approach demonstrates clear application value and substantial potential for widespread adoption, thereby promoting the continuous innovation and advancement of cybersecurity technology within the enterprise environment.

The main contributions of this study are as follows:

(1) Dedicated Staff Contribution to Information Security

- Enhanced Expertise Utilization: Information security specialists play a critical role in the model training and result validation processes; their expertise and experience are essential to the accuracy and practical utility of the models.
- Shortened Response Time: AI tools significantly aid cybersecurity professionals in quickly interpreting threat intelligence, thereby reducing response times and minimizing cybersecurity risks. (Pursuant to cybersecurity incident reporting and response protocols, incidents must be reported within one hour of detection. Assuming an average of one hour per confirmed case for incident response, leveraging AI is estimated to enhance operational efficiency by 20%.)
- Strategic Focus on Policy Decisions: By automating threat detection and situational analysis, AI tools enable security officers to focus their efforts on developing strategic security policies.

(2) Impact of AI-funded security protection

- Automated Threat Identification: Machine learning models automatically identify potential security threats, improving the efficiency and accuracy of threat detection.
- Accelerated Intelligence Analysis: Text mining technology expedites the extraction of key information from threat intelligence, thereby enhancing the overall efficiency of intelligence analysis.
- Visualized Threat Landscape: Text visualization techniques transform complex threat data into intuitive graphical representations that quickly provide security officers with a comprehensive overview of the threat landscape.

4.2 Recommendations and Future Research Directions

(1) Recommendations

This study focused on threat intelligence emails, proposing a method for key feature information extraction based on the subject and content. We constructed a spam detection model by integrating machine learning classifiers, including Decision Tree, Bayesian classification, and Support Vector Machine (SVM). The experimental results demonstrated that the model, particularly when incorporating the TF-IDF feature selection algorithm, exhibits clear advantages in threat intelligence email detection and can effectively improve identification accuracy.

(2) Future Research Directions

To further enhance the performance and utility of threat intelligence email detection models, the following future research directions are recommended:

- Exploration of Advanced Natural Language Processing (NLP) Techniques: More advanced NLP techniques, such as the Transformer architecture, could be applied to key feature information extraction and recognition in threat intelligence emails to improve the model's accuracy and efficiency.
- Incorporation of Diverse Threat Intelligence

Sources: Integrating diverse threat intelligence sources, such as real-time threat feeds, social media data, and user behavior analytics, into model training would enhance the model's generalization capability and robustness.

- Research on Interpretable AI (XAI) Techniques: Investigating XAI techniques, such as LIME or SHAP, is crucial for improving the interpretability of model decisions. This would help security and critical operations engineers better understand the model's operation mechanism and foster greater trust in the system.

References

[1] National Institute of Cyber Security, *N-SOC Development Direction Description*, National Information and Communications Security Monitoring Center. [Online]. Available: https://www.nics.nat.gov.tw/core_business/cybersecurity_defense/N-SOC/

[2] National Institute of Cyber Security, *N-ISAC Fact Sheet*, National Information Sharing and Analysis Center. [Online]. Available: https://www.nics.nat.gov.tw/core_business/information_security_information_sharing/National_Cyber_Security_Information_Sharing_and_Analysis_Center/

[3] Administration for Cyber Security, moda, *Monthly Information Security Network Report: Joint Defense Monitoring Statistics (Jan.–Dec. 2023)*. [Online]. Available: https://moda.gov.tw/ACS/press/report/670

[4] National Institute of Cyber Security (NICS), *Technical Reports: Information Security (Q1–Q4 2023)*. [Online]. Available: https://www.nics.nat.gov.tw/cybersecurity_resources/publications/Technical_Reports/

[5] W.-W. Lin, *Construction of Security Threat Information Application System Based on STIX/TAXII Standard*, M.S. Thesis, Shu-Te University, Kaohsiung, Taiwan, 2020. https://hdl.handle.net/11296/9qafs7

[6] H.-Y. Lin, *Text Mining and Event Study – In the case of mutual fund*, M.S. Thesis, Fu Jen Catholic University, Taipei, Taiwan, 2018. https://hdl.handle.net/11296/dd35zg

[7] W.-C. Hsieh, Y.-H. Hu, Automated classification of cybersecurity incidents using text mining techniques, *Journal of Information Communication and Technology Auditing*, No. 29, pp. 92–101, January, 2014.

[8] J.-H. Seo, E.-M. Park, A study on financing security for smartphones using text mining, *Wireless Personal Communications*, Vol. 98, No. 4, pp. 3109–3127, February, 2018. https://doi.org/10.1007/s11277-017-4121-7

[9] H.-K. Yu, *Using Support Vector Machine and Text Mining For Stock Price Trends Prediction*, M.S. Thesis, Fu Jen Catholic University, Taipei, Taiwan, 2008. https://hdl.handle.net/11296/5ybjxb

[10] Y.-S. Lin, *Research on using R Language for Threat Intelligence Mining, Incident Resolution and Implementation Validation*, M.S. Thesis, Tatung University, Taipei, Taiwan, 2024. https://hdl.handle.net/11296/86bexn

[11] J.-Y. Hau, *News event tracking using text mining*, M.S. Thesis, Tamkang University, New Taipei, Taiwan, 2017. https://hdl.handle.net/11296/z5632v

[12] Administration for Cyber Security, Ministry of Digital Affairs, *Background of the National Information and Communication Security Taskforce (NICST)*, 2024. [Online]. Available: https://moda.gov.tw/ACS/nicst/background/658

[13] D. Jurafsky, J. H. Martin, *Speech and Language Processing* (3rd Edition, Draft), Upper Saddle River, NJ, USA: Pearson, 2023.

[14] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, K. Kochut, A brief survey of text mining: Classification, clustering, and extraction techniques, *arXiv preprint*, arXiv: 1707.02919, July, 2017. https://arxiv.org/abs/1707.02919

[15] A. Singh, J. Prakash, G. Kumar, P. K. Jain, L. S. Ambati, Intrusion Detection System: A Comparative Study of Machine Learning-Based IDS, *Journal of Database Management (JDM)*, Vol. 35, No. 1, pp. 1–25, 2024. https://doi.org/10.4018/JDM.338276

Biographies

Hung-Cheng Yang received a bachelor's degree in traffic management in Taiwan in 1984 and a master's degree in information management from the Central Police University in Taiwan in 2003. He has been studying computer science and Information engineering at Tatung University for a Ph.D. since 2021. Since 1984, he has worked for the police. His research interests include digital evidence and forensics, and traffic accident prevention using big data. (yangyeh5046@gmail.com).

I-Long Lin received a B.S. degree from Central Police University, Taiwan, in 1983 and M.S. and Ph.D. degrees from Tamkang University, National Taiwan University of Science and Technology, Taiwan, in 2002 and 2005. From 1983 to 2011, he worked as a professor at Central Police University, Taiwan. Since 2012 to 2021, he was a professor at Yuanpei University of Medical Technology, Taiwan. Since 2021, he has been a professor at Tatung University. His research interests include digital evidence and forensics, and cybersecurity. (cyberpaul@gm.ttu.edu.tw)

Yu-Shan Lin received a bachelor's degree in Electronic Engineering in 1994 in Taiwan and completed his Master's degree in Computer Science at Tatung University in 2024. From 1995 to 2022, he served as a dedicated cybersecurity specialist for smart healthcare at Luodong Poh-Ai Hospital in Taiwan. Since 2022, he has been working as a telecommunications cybersecurity specialist at Taiwan Mobile Co., Ltd. His research interests include digital evidence and forensics, as well as the development and application of AI in the field of information security. (hinet.lys6103@gmail.com)

Chorng-Ming Chen earned a Bachelor's degree from National Chengchi University in 1995, a Master's degree from the University of Management and Technology, USA, in 2004, and another Master's degree in Computer Science and Information Engineering from Tatung University in 2023. Since 2023, he has been pursuing a Ph.D. at Tatung University. He has held roles such as R&D Manager, CTO, and CISO since 1986. His research interests include intelligent customer service, information security, digital forensics, AI, and LLM. (avenchentw@gmail.com)