Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

AI Based Novel Model for Prediction of Cyber Security Attacks Using KNN Algorithm & XDR

Jadhav S D.[1], Bombade B R.[2]

[1]Research Scholar, [2]Associate Professor, Shri Guru Gobind Singhji Institute of Engineering & Technology, Nanded, India.

Email: [1]2021PCS103@sggs.ac.in, [2]brbombade@sggs.ac.in

Abstract

Cyberspace is treated as a fourth dimension of modern-day warfare apart from land, air and sea. Solutions are developed to provide cybersecurity to computer systems, but every time the attacker tries new methodologies and overcomes the security systems. Such a set of tools and solutions also consists of Log Analysis solutions. It is a proven fact that, Log Analysis helps to predict and prevent cybersecurity attacks. However, very few research attempts have been made regarding the application of Artificial Intelligence to Log Analysis (especially Extended Detection and Response (XDR) Log Analysis). Therefore, in this paper we propose and implement a K-Nearest Neighbors (KNN) algorithm based preventive and predictive system. The K-Nearest Neighbors algorithm is a non-parametric supervised learning algorithm. Extended Detection and Response (XDR) is one of the modern solutions that has the capability to collect and process data from various sub-systems connected in a given network and is an information goldmine from the cybersecurity audit perspective. In this paper, we propose to use the KNN algorithm over the XDR. Therefore, the proposed novel model includes steps such as; Input the data, checking for “missing values” and “duplicate entries”, identifying available “classes” and optimizing them to two or three Major Classes, then performing “label encoding” and creating the “correlation value-based matrix”. Further, we find out “Positive” and “Negative correlation values” and discard the rest values, then select the features which has highest correlation values. Later, we apply the scikit-learn class standard scaler method to scale the features to centre the data around a mean of "0" and a standard deviation of "1." Finally, apply the KNN classifier with Optuna to identify the K-nearest neighbor. This will generate the final output, which will define, whether the given log entry is of “Suspicious Class” or “Not Suspicious Class”. The Suspicious Class XDR log entries will be dealt with separately, as they might indicate a potential risk or incident of compromise (IOC). The proposed novel experiment has been tested on the standard 357icrosoft based GUIDE dataset and a locally generated in-lab dataset. The Microsoft GUIDE XDR data contains 13 million pieces of evidence across 33 entity types, 1.6 million alerts, and 1 million well annotated incidents collected from 6,100 organizations. In both cases, our experimentation has successfully achieved a result of 93.85% accuracy in predicting cybersecurity attacks.

Keywords: Cyber Security, Computer System Logs, Log Analysis, Artificial Intelligence, XDR (Extended Detection and Response), XDR Log Analysis.

DOI: https://doi.org/10.36548/jtcsst.2025.3.004
Received: 03.07.2025, received in revised form: 01.08.2025, accepted: 15.08.2025, published: 26.08.2025

1. Introduction

Cyber security is a major challenge in the 21st century. The world came together through the use of the internet. Millions of people across the globe have connected seamlessly 24/7, 365 days a year. Be it banking or defense, research or entertainment, all are connected through the internet, which has provided the convenience of accessing another person's system in remote ways and carrying out operations without any interruptions. However, there exists a clear and present danger on the internet called cybercrime, which is perpetrated by cyber criminals across the globe.

Organizations and individuals have already lost billions of dollars in cyber security-related frauds. This includes cyber economic fraud, malware attacks, ransomware attacks, romance fraud, online shopping scams, lottery scams, identity theft, digital arrest scams, phishing theft, data theft, cyber stalking and bullying, social engineering attacks, hacking and spear phishing, cyber extortion, cyber blackmailing, cyber grooming, child exploitation, child pornography, coercive sexting, dark web frauds, cryptocurrency frauds, online share market frauds, online lottery scams, email spoofing, credential theft, credit card detail hijacking, personal and private image hacking, hospital/patient medical data theft, online piracy and copyright infringement, cloud data theft, and theft of data from defence research and development organizations. The list is not limited to the above cybercrimes, as cybercrime evolves over time, and every year a new type of cybercrime is observed. Therefore, to prevent such activities, it is necessary to predict cyber-attacks in real time, and an alert can help the system security administrator take immediate action. Predicting cyber security attacks in real time is a significant challenge for the research community. Various models, theories, tools, and solutions have been designed to detect and/or predict cyber security attacks, but not all are up to the mark concerning the changing modus operandi of cyber criminals and evolving cyber-attack threat vectors. Cyber criminals have always tried to exploit the loopholes in existing cyber security tools and solutions. Such loopholes may exist at the software level, hardware level, or firmware level. Various organizations deploy permutations and combinations of a variety of cyber security tools that include intrusion detection systems (IDS), intrusion prevention systems (IPS), firewalls, antivirus software, security information and event management (SIEM), endpoint detection and response (EDR), extended detection and response (XDR), etc. The arrangement of such tools is entirely subject to the respective organization's matrix, requirement analysis of needs and budget availability, awareness of cyber threats and their seriousness, adherence to local government cyber security guidelines, and other such parameters. More or less, every organization tries to keep itself ready to counter the cyber security challenge, but despite having the best cyber security practices, unfortunately, cyber-attacks occur.

1.1 Introduction to KNN Algorithm

As per our problem statement requirements, we believe that out of the various available algorithms, the KNN algorithm is the most suitable for predicting cybersecurity attacks [13 & 14]. The KNN algorithm is also known as K-Nearest Neighbors. The K-Nearest Neighbors algorithm is a non-parametric supervised learning algorithm. In supervised learning, information is labeled and given to the system, which helps it identify patterns and then classify them. KNN is used for both classification and regression. KNN consists of objects or values and classes. KNN assumes that an object either belongs to a specific class, say class "A," class "B," or class "C," and similar ones. When KNN is used for classification, let's say a new object is discovered; then the plurality vote of its neighbors helps to classify which class the object

belongs to. If the feature set of an object represents varying features that do not help correlate it to any nearest neighbor, then the feature set data is normalized, which in turn helps to classify the object. Highly varying feature set data may expand the boundaries of the class and reduce noise, but it will distinguish the boundaries between the classes. Classification of the object is done using "majority voting" by nearest neighbors, but problems occur when class data distribution is skewed. In such cases, if any class has a majority of objects in it, then it dominates the selection of the new object's class. Object classification can be improved by applying specialized algorithms like the Large Margin Nearest Neighbor algorithm or Neighbourhood Component Analysis algorithm. Figure 1 is the best example of the KNN algorithm used for classification. As per Figure 1, in the above example, one can observe that the test sample is the black dot (placed inside the innermost circle) and needs to be classified either into the red squares or green triangles. In the innermost circle, there are 3 green triangles and 2 red squares; therefore, according to the KNN algorithm, for K = 5, the black dot is assigned to the green triangles. This is because there are only 2 red squares, and their total number is less than 3 green triangles. Similarly, for the outer dashed circle, for K = 10, according to KNN algorithm principles, the black dot is assigned to the 7 red circles (against the 3 green triangles).

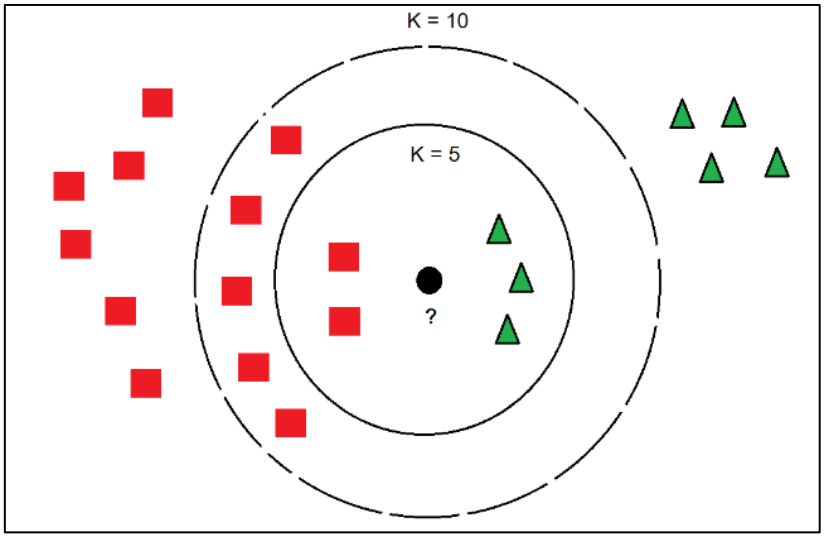

Figure 1. KNN Classification Example for K = 5 and K = 10 Values Respectively

1.2 Introduction to XDR

One of the most recent advanced solutions for cybersecurity is considered to be XDR, which collects and analyzes log data from multiple security layers such as networks, servers, cloud servers, email servers (SMTP), end-point machines, etc. Various stages of XDR log collection and response are shown in Figure 2. XDR has feature-rich columns that contain alerts, suspicious events, critical attention events, incidents of compromise details, triage-annotated incidents, unique identifiers, etc., that are recorded as either true positives, false positives, or neutral. In short, XDR is a unified, improved automated threat detection and response solution. The objective of our research is to use XDR logs and apply the KNN algorithm to enhance the prediction method for cybersecurity attacks in real time. Three major differences between our proposed research work and previously conducted research are as follows: a) In previous research, predictions were made by compiling logs from individual security systems, such as those based on firewall logs, IDS logs, or IPS logs; b) No collective

approach was provided to compile all available logs collected from multiple cybersecurity systems and network layers; and c) Very few approaches have been proposed for real-time prediction of cybersecurity attacks using the KNN algorithm, particularly utilizing XDR logs.

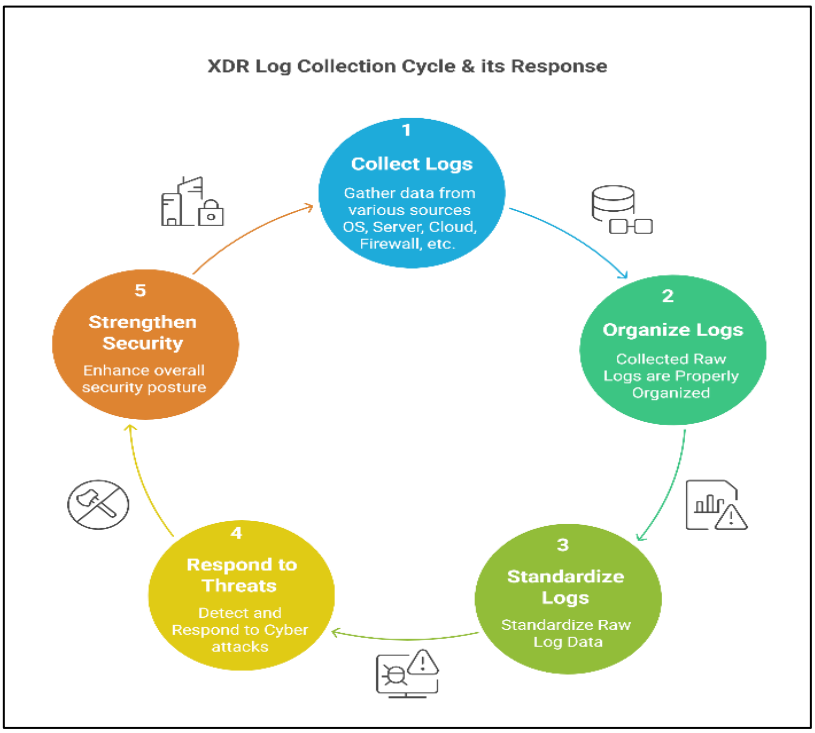

Figure 2. XDR Formal Steps of Log Collection & Its Response to Cyber-Attack

1.3 Research Problem Statement

Cybercrime across the globe is increasing rapidly, and billions of dollars' worth of data and infrastructure are at stake. The world needs to address such problems through continuous research. Prediction of cybercrime through the compilation of various logs and records received from computer systems, digital subsystems, servers, etc., is possible, but it is a challenging task. The various cybersecurity systems developed, such as firewalls, antivirus, IDS, IPS, SIEM, EDR, XDR, and others, provide cybersecurity at various levels and belong to their respective proprietors. Few of the above systems collect and monitor logs for detecting possible cybersecurity attacks. However, despite that, cybersecurity attacks still occur. At present, it is observed that log monitoring and processing security systems have various issues, such as poor log optimization algorithms, slow log analysis rates, a higher rate of false positive identification than correct identification of anomalies and incidents of compromise (IOCs), and one of the major issues is that no collective approach is given to the compilation of all the available logs. The majority of research work has focused on the analysis of individual subsystem logs, such as Windows event log analysis, IDS log analysis, firewall log analysis, or IPS log analysis, etc.

Therefore, our research problem statement focuses on the prediction of such cybersecurity attacks through effective compilation of all available logs, especially all the logs collected by modern XDR systems from various systems, digital subsystems, and cybersecurity tools. This will help advance research by providing a collective log compilation for comprehensive analysis and prediction of cybersecurity attacks. It is well said that prevention is better than cure. Therefore, it is better to avoid cybersecurity attacks through prediction, and hence we propose to do this using the KNN algorithm. The KNN algorithm will help resolve

the problem statement, as it has the ability to address various classes created by the XDR logs. Furthermore, it not only helps to define the boundary conditions of classes but also calculates the distances among the nearest k-values for any given log entry. From a technical perspective, the time and space complexity for the KNN algorithm is as follows: Training Time: O(1), Training Space: O(nd), Prediction Time: O(nd), Prediction Space: O(nd).

1.4 Current Solutions

Presently, there are various other solutions; this includes IDS, IPS, SIEMs, and others. However, each tool acts independently. Similarly, EDR (Endpoint Detection and Response) and MDR (Managed Detection and Response) both have limitations, as they are restricted to endpoint visibility, which means they cannot address cloud-based threats, email-based threats, etc. Additionally, they provide very little customization compared to XDR systems.

1.5 Limitations

The various limitations of the research work are that if the policy opts not to include certain computer systems or sets of digital devices under the security umbrella of the XDR system, then the collection of logs will be limited (though this is a rare case). Additionally, when the dataset size approaches infinity, the two-class KNN algorithm produces errors, and such error rates are mostly twice that of the Bayes algorithm's error rate. Furthermore, the proposed research requires very high-end computing power and memory.

2. Related Work

This section provides an overview of various methods and systems developed to predict cybersecurity attacks using multiple artificial intelligence-based machine learning algorithms, including the KNN algorithm. The main objective of this section is to understand key findings and highlights related to the research contributions made using the KNN algorithm.

M. O. Adebiyi et al. [15] proposed a method to use KNN and RNN in combination to detect intrusions over the network. The method uses KNN for identifying k-nearest neighbors, while RNN captures the temporal dependencies present in the sequential data. The method has used the Intrusion Detection System (IDS) dataset for training purposes. G. Shanmuga Priya et al. [16] proposed a model where KNN is used along with the Gaussian Mixture Model (GMM). They initially applied the GMM model, and the post-processing step involved neglecting unwanted noise and objects using GMM filters and morphological operations. Later, statistical features were extracted for each object. Such objects are then classified and labeled as either anomalies or normal events using the KNN classifier. The dataset used in the experimentation is from NLC India.

Lin Liu et al. [17] explained their model in which KNN is used for the classification of outsourced cloud data. The authors explained how they used a set of building blocks such as secure sorting, secure minimum and maximum number finding, secure frequency calculation, and others to design the proposed system using KNN for better classification and threat detection over the cloud system. M. Agarwal et al. [18] explored the potential of KNN based on network-related attacks. The authors emphasized exploring the possibility of labeling cybersecurity attacks using KNN at the network level. The primary objective of the research was to monitor and track network security attacks using the KNN algorithm.

M. M. Hassan et al. [19] proposed a methodology for identifying phishing attacks using a stacking ensemble (a combination of one-to-many machine learning algorithms including KNN, Naïve Bayes, Extreme Gradient (XG) Booster Classifier, Support Vector Machine (SVM), and others). A. Abdulboriy and J. S. Shin [20] developed a method where the KNN algorithm, Softmax Regressor, and Adaptive Random Forest classifier are jointly applied to the IDS data to detect anomalies. The combination of more than one machine learning algorithm aims to increase accuracy and reduce false alarm rates.

Similarly, N. Kumar et al. [21] proposed a method in which they used various algorithms such as Logistic Regression, Random Forest, Feedforward Neural Networks (FNN), K Nearest Neighbors (KNN), and Long Short-Term Memory (LSTM), among others, to develop a type of Network IDS system to detect intrusions. The authors proposed the hypothesis that if a set of more than two algorithms is used in designing the system, it will increase precision, accuracy, and reduce the false acceptance rate. There are various other methods where KNN is used in the field of cybersecurity, but no one has ever explored its full potential along with the XDR dataset. In short, KNN is used in combination with other machine learning algorithms for the classification of network intrusion-based attacks, but KNN has never been applied to the XDR dataset for the prediction of cybersecurity attacks. The reason for selecting the K-Nearest Neighbor algorithm is that it is a non-parametric supervised learning algorithm, which means it can handle logs of various kinds of subsystems and can define and manage a multi-class environment for a variety of XDR logs. Additionally, apart from the field of computer science, the KNN algorithm is also used in various other fields such as electronics, mechanical engineering, medical science, information systems, etc. [22, 23, 24, 25, and 26].

3. Proposed Work

This section describes in stepwise manner, how the proposed novel model is used for prediction of cyber-security attacks. Table 1 consists of Major Steps involved in the proposed novel model.

Table 1. Major Steps

S.No	Steps
1	Input Log Dataset Selection, Indexing & Cleansing
2	Define Computational Parameters & Classes
3	Label Encoding & Map Correlation Matrix
4	Feature Selection & Feature Scaling
5	Apply KNN Principles and Optuna
6	Train the Model

In the following sections, all six major steps outlined in Table 1 will be explained thoroughly. For a better understanding, the flowchart of the overall architecture is provided in Figure 3, and it is synchronized with Table 1.

Step 1: Input Dataset Selection, Indexing & Cleansing

Microsoft-based GUIDE dataset 2024 is one of the largest publicly available collections of real-world cybersecurity incidents, which contains 13 million pieces of evidence across over 33 entity types, including 1.6 million alerts and 1 million well-annotated incidents collected from 6,100 organizations [27, 28]. This dataset was collected by integrating Microsoft-developed Co-pilot for Security Guided Response (CGR) into the Microsoft Defender XDR product deployed worldwide. This dataset contains evidence, alerts, and incidents that are recorded as either true positive, false positive, or being positive.

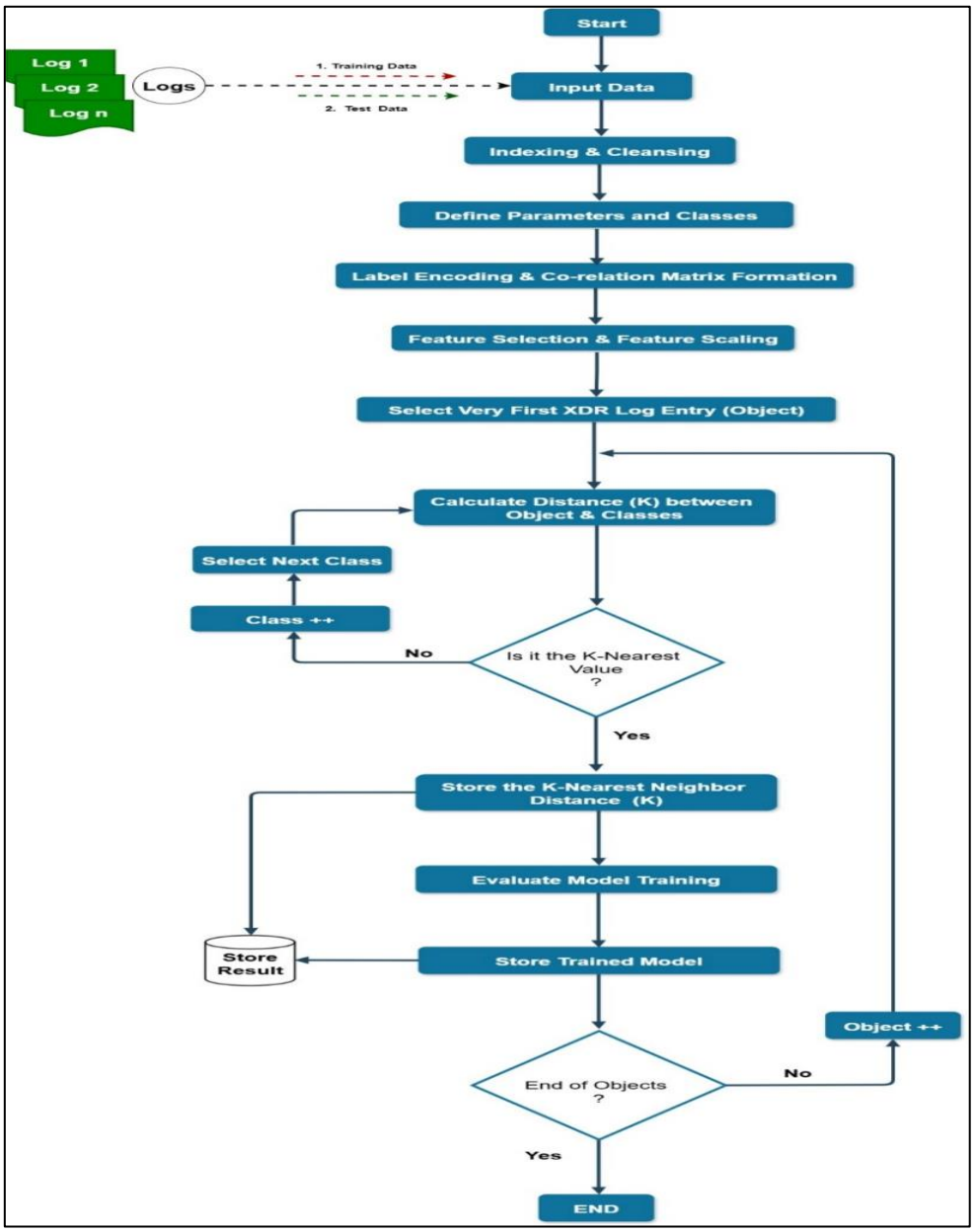

Figure 3. Flow-Chart of Overall Architecture of Proposed Novel Model

One more reason for this database selection is that the dataset contains 44 feature-rich columns, labels, and unique identifiers across 1 million triage-annotated incidents. The dataset is divided into 70% for the training dataset (consisting of 9,516,837 row entries) and 30% for

the test dataset (consisting of 4,147,992 row entries). Also, in order to maintain privacy, the GUIDE project has ensured that sensitive data fields are pseudo-anonymized using the SHA-1 hashing technique without affecting the other fields/values in the dataset. Additionally, we have used our own lab dataset to test it against the standard dataset. The results are quite similar; the only difference we found is in the number of incidents and anomalies available in both datasets. Furthermore, the received dataset is indexed in ascending date-wise order and then cleansed. In the data cleansing process, all the columns and rows are first scanned, and then the row-columns that have null values (or missing values) and duplicate entries are removed. This is done because null values and duplicate entries in any row or column add no value to the results; rather, they occupy memory and waste computation time.

Step 2: Define Computational Parameters & Classes

Once the Input Log dataset is cleansed, the next step is to define the parameters based on which the Logs will be processed. First check whether the input log file is empty or not.

$\emptyset \rightarrow$ Check if the given log file is empty or null.

$\Theta \rightarrow$ Number of log files taken as input. (Where $\Theta \geq 1$)

Let,

$\Delta \rightarrow$ Number of successive entries of events (in terms of logs) in each log file ($\Delta 1, \Delta 2, .. \Delta n$).

$\delta \rightarrow \delta 1, \delta 2, \delta 3, \ldots\ldots.. \delta n$ be the various successive suspicious events found in the given respective log file (detected during sequential log file processing).

$\Psi \rightarrow \Psi 1, \Psi 2, \Psi 3, \ldots\ldots\ldots.. \Psi n$,be the various successive malicious events or incidents of compromise found in given log file (s).

Now, define the classes on the basis of which the dataset will be categorized, Let;

$\Omega \rightarrow$ be the set of series of classes used for the classification of logs. For example;

$\Omega\ 1 \rightarrow$ Non – Suspicious log entries

$\Omega\ 2 \rightarrow$ Suspicious log entries

$\Omega\ 3 \rightarrow$ Alert level log entries

$\Omega\ n \rightarrow$ Malicious log entries

Step 3: Label Encoding & Co-relation Matrix

The next step is label encoding. All the classes need to be processed with label encoding. Label encoding helps to simplify the data, makes implementation easier, and helps KNN algorithm in calculating the distance between data points to create the correlation matrix. In the label encoding process, all the classes will be assigned computational weights/values of either “0”, “1”, or “-1” as defined below;

$W = 0 \rightarrow$ Zero stands for “No Co-relation”

$W = 1 \rightarrow$ One stands for “Positive Co-relation”

W = -1 → Minus One stands for "Negative Co-relation"

(For example, the Suspicious class will be assigned the value "1," and the Non-suspicious class will be assigned the value "0.")

We are least interested in a correlation value of "0," as it signifies that there is no relationship between the variables. Therefore, from our experimental perspective, we are interested in correlation matrix values between 1 and -1, as they represent both positive and negative correlations among variables, respectively.

Step 4: Feature Selection & Feature Scaling

The Microsoft GUIDE dataset has 44 feature-rich columns, but not all of the 44 feature-rich columns are useful. Once correlation matrix values are obtained, there is a need to identify those features (out of the 44 feature-rich columns) that have the highest correlation values. This is because only the features with the highest correlation values can help to find out the "k-values." After feature selection, feature scaling is done. The feature scaling process centers the data around zero, standardizes the variability, and further helps to increase the accuracy of the final result.

Step 5: Apply KNN Principles and Optuna

After feature selection and scaling, the KNN classifier with Optuna is applied to the dataset to find the best neighbors (those with the highest matrix correlation value) As per figure 4, let us assume,

n → is set of all neighbors. Say n = {1, 2, 3, 4, 5, ... n}

i → is the i^{th} nearest neighbor

Weighted Nearest Neighbor Classifier is defined as;

C_n^{wnn} → denotes the weighted nearest classifier with weight $\{w_{ni}\}^n$ i = 1

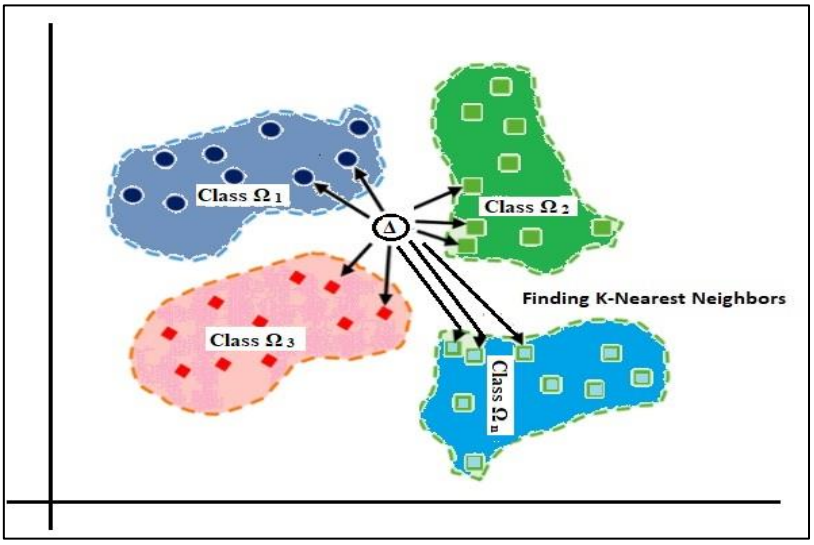

Figure 4. Finding K- Nearest Neighbor in KNN \Algorithm

Such closest neighbor classifier ensures that, the error rate won't be worse than twice the Bayes Error Rate (R), in such a way that;

R → Bayes algorithm error rate

RKNN → KNN algorithm error rate

M → Total Number of classes

Then, Optuna uses Bayesian optimization (Tree-structured Parzen Estimator) to select the hyperparameter combinations that help to improve the overall accuracy of the final result by finding the best neighbors. Furthermore, for multi-class KNN classification, the upper bound error rate is defined as;

$$\mathrm{R} \leq \mathrm{RKNN} \leq \mathrm{R}\left\{2-\left\{\frac{\mathrm{MR}}{\mathrm{M}-1}\right\}\right\}$$

However, there exists more risk if we find asymptotic growth of the class distribution in the given dataset. This it should be addressed, such that;

$R_R(C_n^{wnn}) - R_R(C^{Bayes}) = (B_1 S_n^2 + B_2 t_n^2)(1 + \mathrm{O}(1))$ → *for Constant* B_1 *and* B_2

Where; $S_n^2 = \sum_{i=1}^{n} W_{ni}^2$

and,

$$t_n = n^{-2/d} \sum_{i=1}^{n} W_{ni} \left((i^{1+2/d}) - (i-1)^{1+2/d} \right)$$

The optimal weighting scheme $(W_{ni}^*)_{i=1}^{n}$ that keeps the above two terms in balance is given as follows;

Set K^* such that; $\mathrm{K}^* = \left\lfloor Bn^{\frac{4}{d+4}} \right\rfloor$ and

$$W_{ni}^* = \frac{1}{K^*}\left[1 + \frac{d}{2} - \frac{d}{2k^{*2/d}} \{ i^{1+2/d} - (i-1)^{1+2/d} \} \right]$$

$$for\ i = 1, 2, \ldots, k^*\ and\ W_{ni}^* = 0$$

$$for\ i = k^* + 1, \ldots, n$$

Therefore, as per the above KNN method, now select the very first log entry Δ1 and then calculate the distance between the given object and the available classes (Ω 1, Ω 2, Ω 3,…. Ωn).

Let us consider the log entry Δ1 as the initial log entry/object and therefore the Weighted Nearest Neighbor Classifier assigns weight "1" (ONE) to nearest neighbor and all other neighbors are assigned weight "0" or "-1" such that;

$$\mathrm{Wni} = \sum_{i=1}^{n} \mathrm{Wni} = 1$$

Extend this calculation to the boundaries of each of these classes. . Calculate the distance for all such log entries (Δ) concerning the given class (Ω 1, Ω 2, Ω 3,….Ωn). Check all such classes and calculate the distance (k). The value of distance (k) defines whether the object belongs to which class from all the available class. Once the appropriate nearest neighbor class is found, then the object is assigned to that respective class.

Sort the above function for all values of Δ till EOF and respectively assign each one to their respective classes Ω 1, Ω 2, Ω 3,…. Ωn accordingly.

Step 6: Train the Model

Store the K-nearest neighbor value and the overall learnings. In other words, this means storing all the final k-values assigned to all log entries (Δ) and the respective class (Ω 1, Ω 2, Ω 3,…. Ωn) assigned to each of those log entries. This will further help to train the novel model. A well-trained model can be mature enough to assist in the early prediction of cybersecurity attacks through each new log entry it find in the system.

4. Results and Discussion

For implementation, we used Python 3.0 as it has a rich set of libraries and the latest security updates. Therefore, the experiment was conducted with various standard databases available and our local lab's in-house/institute-created database. Since the programming code and database require high computational power and RAM to compute and generate results, we used both our institute's local servers and the Google Colab Pro+ paid facility. From Figure 5 (a), one can observe that the code is compiled on Google Colab. The window shows the initial code written for the project titled "AI-Based Novel Method for Prediction of Cyber Security Attacks Using KNN Algorithm" and various library inclusions.

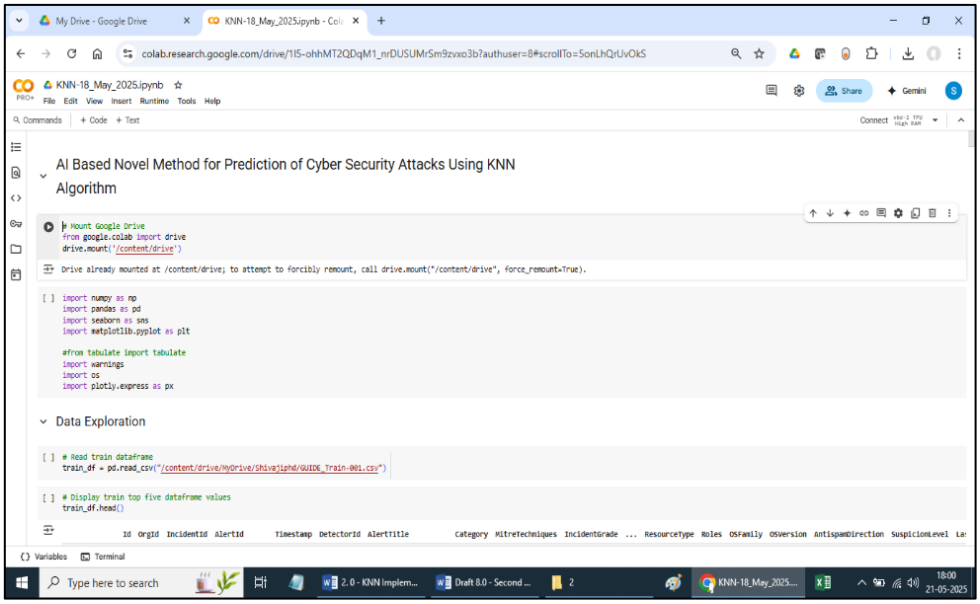

Figure 5 (a). Implementation Snapshot Showing Project Title and Libraries Inclusion in Google Co-Labs

As per the steps mentioned in Table 1, we followed the proposed novel model execution. The Microsoft-based GUIDE dataset is divided into 70% training dataset (consisting of 9,516,837 row entries) and 30% test dataset (consisting of 4,147,992 row entries). It has 44 feature-rich columns, as mentioned below:

Id, OrgId, IncidentId, AlertId, Timestamp, DetectorId, AlertTitle, Category, MitreTechniques, IncidentGrade, ActionGrouped, ActionGranular, EntityType, EvidenceRole, Roles, DeviceId, DeviceName, Sha256, IpAddress, Url, AccountSid, AccountUpn, AccountObjectId, AccountName, NetworkMessageId, EmailClusterId, RegistryKey, RegistryValueName, RegistryValueData, ApplicationId, ApplicationName, OAuthApplicationId, ThreatFamily, FileName, FolderPath, ResourceType, OSFamily, OSVersion, AntispamDirection, SuspicionLevel, LastVerdict, CountryCode, State, City.

Out of 44 feature-rich columns, there exist a certain number of rows/columns that are empty or contain null values. For the proposed experimentation, we recommend removing all such columns that have more than 40% null or empty values. There are 8 columns that have more than 40% null values, and they are:

MitreTechniques', 'ActionGrouped', 'ActionGranular', 'Roles', 'EmailClusterId', 'ThreatFamily', 'ResourceType', and 'AntispamDirection'.

Since these 8 columns have been removed, we now have a total of 36 feature-rich columns available for computation. Similarly, if any missing values or duplicate entries are present in the given dataset, those entries also need to be checked and removed before the actual computation starts. For the Microsoft GUIDE training dataset, we found that there are a total of 51,340 missing values and 119,477 duplicate entries. All such entries were removed from computation. Now, the total number of row entries present in the Microsoft dataset is 93,460,20. This action is part of the data cleansing process. Further, we found few major classes in given Microsoft Guide training dataset; *Not Suspicious, Suspicious, Malicious, NoThreatFound, two classes of DomainPII.*

But initially, for simplification, we opted for only two major classes: Not Suspicious and Suspicious. This is done by merging the *NoThreatFound* class with the Not Suspicious class and merging the Malicious class with the Suspicious class. The other two classes of *DomainPII* are discarded. This is shown in Figure 5(b). Now, the total number of row entries present in the Microsoft Dataset is 93, 45, 829. Once the classes are defined, the next step is to perform label encoding.

	count
Not Suspicious	7204529
Suspicious	1378546
Malicious	389538
NoThreatsFound	373216
DomainPII_50d8b4a941c26b89482c94ab324b5a274f9ced66	128
DomainPII_9207384283ce115db5a590dd9ca5de21e5e99df2	63

	count
Not Suspicious	7911626
Suspicious	1434394

Figure 5 (b). GUI Showing Input File Contains Classes; Not Suspicious, Suspicious and Others.

In the label encoding process, the suspicious class will be assigned the value “1” and the Not Suspicious” class will be assigned “0” value. Label encoding helps to simplify the data, makes implementation easier, and assists the KNN algorithm in calculating the distance between data points.

After the label encoding process is completed, the input dataset file shows correlation matrix values. A correlation matrix summarizes the pairwise correlations between variables present in the Microsoft Guide dataset. The formal diagrammatic representation of the matrix correlation values of the given input file is shown in Figure 5 (c) and Table 2. It illustrates the strength and direction of linear relationships between all pairs of features. It helps to understand the changes in variables, any existing data dependencies, and is also useful for feature selection. The matrix correlation coefficient value ranges from -1 to 1, as shown below:

0 → Zero stands for “No Co-relation”

1 → One stands for “Positive Co-relation”

-1 → Minus One stands for “Negative Co-relation”

Here, the least interesting correlation values are “0” and “-1,” as the initial “0” specifies that there exists no relation between the variables, and the latter “-1” specifies that there exists a negative correlation. Similarly, the correlation value “1” specifies positive correlation; in other words, it is the correlation of the feature itself (the symmetry of the respective row/column with itself). Therefore, from our experimental perspective, we are interested in matrix correlation coefficient values that lie between “0” and “1.”

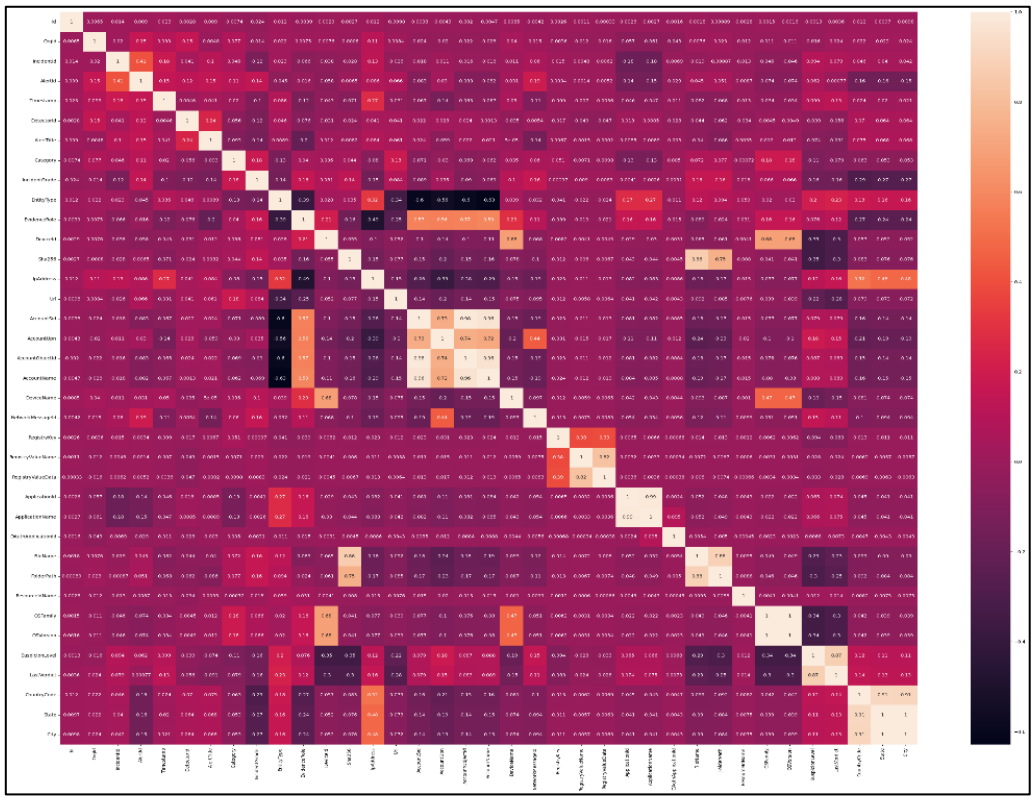

Figure 5(c). GUI Showing the Matrix Correlation Coefficient Values for Given Dataset File

Table 2. Selection of 10-Highest Correlation Coefficient Values

	Id	**OrgId**	**IncidentId**	**AlertId**	**DetectorId**	**Alert Title**	**Category**	**Entity Type**	**Timestamp**	**Network messageid**
Id	1	0.00651	0.014289	0.009031	0.002814	0.009041	-0.00736	0.011847	0.022524	-0.004207
OrgId	0.006510	1	0.019718	0.154481	0.152091	-0.004752	0.076763	0.021925	0.093288	0.015366
IncidentId	0.014289	0.019718	1	0.410135	0.040751	0.103429	0.045989	0.023363	0.175401	0.059801
AlertId	0.009031	0.154481	0.410135	1	0.124106	0.145167	0.109127	-0.045403	0.153639	0.188993
DetectorId	0.002814	0.152091	0.040751	0.124106	1	0.23923	-0.05611	0.046374	0.004572	-0.105955
Alert Title	0.009041	-0.004752	0.103429	0.145167	0.23923	1	-0.09309	0.008874	0.041247	0.00536
Category	-0.007357	0.076763	0.045989	0.109127	-0.056111	-0.093086	1	-0.12955	0.019555	-0.142218
Entity Type	0.011847	0.021925	0.023363	-0.045403	0.046374	0.008874	-0.12955	1	0.086475	0.060476
Timestamp	0.022524	0.093288	0.175401	0.153639	0.004572	0.041247	0.019555	0.086475	1	0.160897
Network messageid	-0.004207	0.015366	0.059801	0.188993	-0.105955	0.00536	-0.14222	0.060476	0.160897	1

Now, the next step is feature selection. Out of 44 columns, we have already discarded 8 columns. Now, we have a total of 36 columns available. Out of these 36 feature-rich columns, the highest 10 feature-rich columns were finalized using the mutual_info_classif function. They are:

'DetectorId', 'AlertTitle', 'AlertId', 'IncidentId', 'Category', EntityType', 'OrgId', 'Timestamp', 'Id', and 'NetworkMessageId'.

This is shown in figure 5(d). These 10 highest Co-relation parameters will further help to find out the k-values.

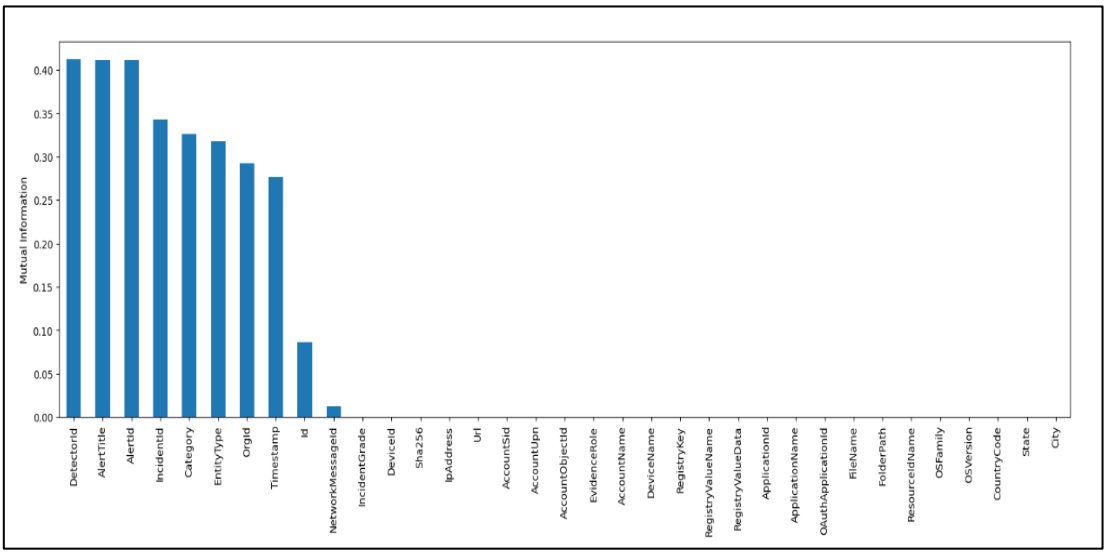

Figure 5(d). GUI Showing 10- Highest Feature Rich Columns Found in Given Dataset File.

Now, the next process is to do the feature scaling. The importance of feature scaling is that if it is not done, features with different scales disproportionately influence KNN learning and the final output. There are various methods for feature scaling (Min-Max Scaler, Max-Abs Scaler, Robust Scaler, Scikit-learn, etc.). For feature scaling, we applied the Scikit-learn class Standard Scaler. It narrowed down the wider scale of the correlation matrix to a mean of 0 and a standard deviation of 1 (this process is also known as z-score normalization). This further helps to increase the accuracy of the final result. Furthermore, we applied the KNN Classifier with Optuna over the training dataset to find the best neighbors (which have the highest correlation value). Optuna uses Bayesian optimization (Tree-structured Parzen Estimator) to select the hyperparameter combinations that help improve the overall accuracy of the final result by finding the best neighbors. In a serial manner, we first processed the Microsoft GUIDE training data sample and then the test data samples. We later used the locally created in-house dataset. The performance and comparison of results for both the training data and test datasets are shown in Table 3.

Table 3. Final Result

Data	No. of Log Entries	Result (approx.)
Microsoft Guide Training Data	93,46,020	97.07 %.
Microsoft Guide Test Data	41,47,992	93.85 %
Local Training Data	93,46,020	80.05 %.
Local Test Data	41,47,992	76.42 %
Final Result		93.85 %

Further the comparison of each training and test dataset with respect to its Accuracy, Precision, Recall and F1 Score is shown in figure 6.

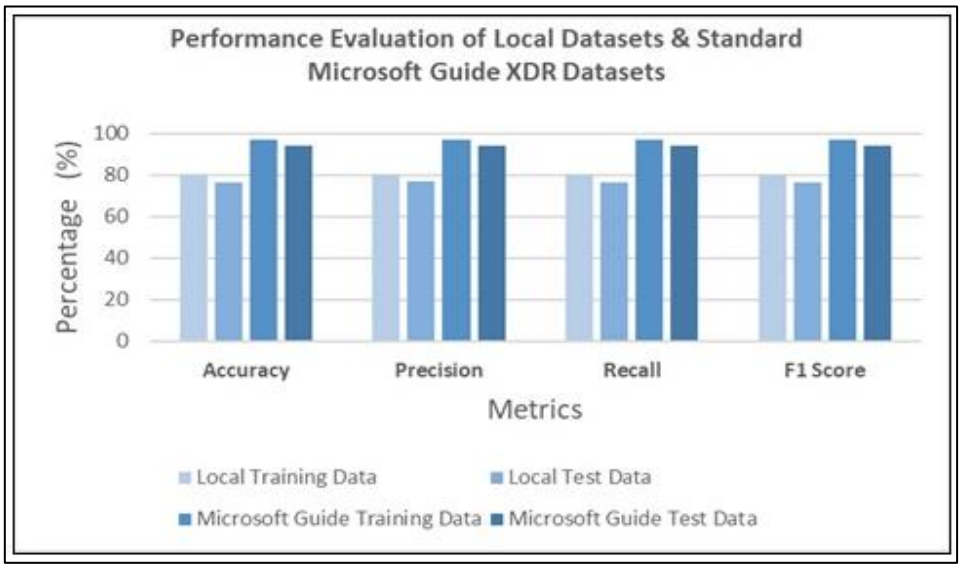

Figure 6. Performance Evaluation on Local Dataset & Standard Microsoft Guide XDR Dataset

As the local in-lab generated dataset samples are non-standard, one cannot completely rely on them. Therefore, the standard Microsoft Guide Test dataset, which consists of 41,479,992 row entries, will be considered the final result with 93.85% accuracy. From the above results, it is now crystal clear that the proposed algorithm, after processing the training data and then the test data, has achieved an overall accuracy of 93.85% in finding the k-nearest neighbor value, which helped to define whether the input log is of the "Suspicious Class" or "Not Suspicious Class." We have conducted experiments with both standard datasets and our locally available lab datasets. In the conducted experiment, a set of logs was provided in a stepwise manner to the system. This helped us to observe the changing status and behavior of the KNN algorithm, various parameters, and further optimize and improve it to achieve higher accuracy in prediction.

5. Conclusion

Cybersecurity attacks have changed the landscape of the global internet. An unprecedented number of cybersecurity attacks on industry, academic and research institutes, and the defense sector have been observed in recent years. Trillions of dollars' worth of losses in terms of financial capital and data theft have occurred. There exists a variety of cybersecurity tools, including firewalls, spam filters, antivirus software, IDS, IPS, SIEMs, EDR, XDR, and others used for the prevention of cybersecurity attacks. Our close observation found that every time cybercriminals attack the digital infrastructure, they create a digital footprint. Such footprints are available in the form of logs and other records. In this paper, we propose a novel model for the prediction of cybersecurity attacks in real time. In the proposed novel model, the KNN principles are applied over the standard Microsoft Guide XDR dataset. The novelty of the proposed system is that XDR is itself a novel system developed by Microsoft, Palo Alto, and other vendors. XDR was recently launched, just a few years ago, and its workings are highly complex. The output of the XDR system is too complex to understand, and no one has ever applied the KNN algorithm over the XDR logs. The proposed novel model comprises six major steps, one of which is the application of the KNN Classifier with Optuna over the XDR dataset. The KNN Classifier with Optuna was initially applied over the training dataset

(consisting of 9,346,020 log entries). At this stage, every single log correlation coefficient or feature is successfully defined, and based on the respective k-value parameter, the distance between the variable and all the available classes is measured. Then, the variable is assigned to the class which has the lowest k-value. Here, the training data samples have successfully predicted the cybersecurity attack with 97.07% accuracy. Meanwhile, the test dataset consisting of 4,147,992 log entries was tested, and the proposed novel model has successfully predicted the k-nearest neighbor class with 93.85% accuracy.

References

[1] Vergara Cobos, Estefania and Cakir, Selcen. 2024. A Review of the Economic Costs of Cyber Incidents. Washington, DC: World Bank.

[2] Sunny, A. "A study on financial cyber-crimes, trends, patterns, and its effects in the economy." Addict Criminol 7, no. 1 (2024): 186.

[3] National Crime Record Bureau (NCRB) Annual Report on Crimes in India 2022. Volume I. Ministry of Home Affairs, Government of India.

[4] Antonescu, Mihail, and Ramona Birău. "Financial and non-financial implications of cybercrimes in emerging countries." Procedia Economics and Finance 32 (2015): 618-621.

[5] Fujimoto, Mariko, Wataru Matsuda, and Takuho Mitsunaga. "Detecting attacks leveraging vulnerabilities fixed in MS17-010 from Event Log." In 2019 IEEE Conference on Application, Information and Network Security (AINS), IEEE, 2019, . 42-47.

[6] Berlin, Konstantin, David Slater, and Joshua Saxe. "Malicious behavior detection using windows audit logs." In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, 2015, 35-44.

[7] Dwaraki, Abhishek, Shachi Kumary, and Tilman Wolf. "Automated event identification from system logs using natural language processing." In 2020 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2020, 209-215.

[8] Visoottiviseth, Vasaka, and Vatcharanun Moonkhaen. "A centralized system for detecting attacks from windows event logs." In 2023 International Electrical Engineering Congress (iEECON), IEEE, 2023, 367-371.

[9] Fujimoto, Mariko, Wataru Matsuda, and Takuho Mitsunaga. "Detecting abuse of domain administrator privilege using windows event log." In 2018 IEEE Conference on Application, Information and Network Security (AINS), IEEE, 2018, 15-20.

[10] Baráth, Július. "Optimizing windows 10 logging to detect network security threats." In 2017 Communication and Information Technologies (KIT), IEEE, 2017, 1-4.

[11] Dwyer, John, and Traian Marius Truta. "Finding anomalies in windows event logs using standard deviation." In 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, IEEE, 2013, 563-570.

[12] Garcia, Karen A., Raul Monroy, Luis A. Trejo, Carlos Mex-Perera, and Eduardo Aguirre. "Analyzing log files for postmortem intrusion detection." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42, no. 6 (2012): 1690-1704.

[13] Sabry, Fouad. Master Algorithm: Fundamentals and Applications. Vol. 166. One Billion Knowledgeable, 2023.

[14] Sun, Jingwen & Du, Weixing & Shi, Niancai. (2018). A Survey of kNN Algorithm. Information Engineering and Applied Computing. 1. 10.18063/ieac.v1i1.770.

[15] Adebiyi, Marion O., Oladayo G. Atanda, Chidinma Okeke, Ayodele A. Adebiyi, and Abayomi A. Adebiyi. "Network intrusion detection using K-nearest neighbors (KNN) and recurrent neural networks (RNN)." In 2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG), IEEE, 2024, 1-8.

[16] Priya, G. Shanmuga, M. Latha, K. Manoj, and Siva Prakash. "Unusual Activity And Anomaly Detection In Surveillance Using GMM-KNN Model." In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), IEEE, 2021, 1450-1457.

[17] Liu, Lin, Jinshu Su, Ximeng Liu, Rongmao Chen, Kai Huang, Robert H. Deng, and Xiaofeng Wang. "Toward highly secure yet efficient KNN classification scheme on outsourced cloud data." IEEE Internet of Things Journal 6, no. 6 (2019): 9841-9852.

[18] Agarwal, Muskan, Kanwarpartap Singh Gill, Rahul Chauhan, Akanksha Kapruwan, and Deepak Banerjee. "Classification of network security attack using KNN (K-nearest neighbour) and comparison of different attacks through different machine learning techniques." In 2024 3rd International Conference for Innovation in Technology (INOCON), IEEE, 2024, 1-7.

[19] Hassan, Md Mahedi, Ahsan Ullah, Anup Chakraborty, Nurunnabi Sarker, and Bikash Kumar Saha Roy. "Enhancing The Cyber Security Using Ensemble Stacking Model For Phishing Sites Detection With Hyperparameter Tuning." In 2024 27th International Conference on Computer and Information Technology (ICCIT), IEEE, 2024, 1809-1814.

[20] Abdulboriy, Alimov, and Ji Sun Shin. "An incremental majority voting approach for intrusion detection system based on machine learning." IEEE Access 12 (2024): 18972-18986.

[21] Kumar, Naween, Supragya Sharma, Sahil Gupta, Sajal Jain, and Sujal Singh. "Harnessing AI for Cybersecurity: Performance Evaluation of Machine Learning Techniques in Intrusion Detection Systems." In 2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT), IEEE, 2025, 693-698.

[22] Vieira, João, Rui P. Duarte, and Horácio C. Neto. "kNN-STUFF: KNN streaming unit for Fpgas." IEEe Access 7 (2019): 170864-170877.

[23] Kaur, Manbir, Chintan Thacker, Laxmi Goswami, Thamizhvani TR, Imad Saeed Abdulrahman, and A. Stanley Raj. "Alzheimer's disease detection using weighted KNN classifier in comparison with medium KNN classifier with improved accuracy." In 2023

3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), IEEE, 2023, 715-718.

[24] Zhao, Puning, and Lifeng Lai. "Analysis of knn density estimation." IEEE Transactions on Information Theory 68, no. 12 (2022): 7971-7995.

[25] Xing, Wenchao, and Yilin Bei. "Medical health big data classification based on KNN classification algorithm." Ieee Access 8 (2019): 28808-28819.

[26] Tu, Bing, Jinping Wang, Xudong Kang, Guoyun Zhang, Xianfeng Ou, and Longyuan Guo. "KNN-based representation of superpixels for hyperspectral image classification." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 11, no. 11 (2018): 4032-4047.

[27] Microsoft Guide Dataset details; https://www.kaggle.com/datasets/Microsoft/microsoft-security-incident-prediction/data?select=GUIDE_Train.csv doi.org/10.34740/kaggle/dsv/8929038

[28] Microsoft Guide Dataset details; Scott Freitas et al., Cornell University Revised version V4, Nov 2024, https://arxiv.org/abs/2407.09017