Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

Customized K-nearest neighbors' algorithm for malware detection

Mosleh M. Abualhaj[a*], Ahmad Adel Abu-Shareha[b], Qusai Y. Shambour[c], Adeeb Alsaaidah[a], Sumaya N. Al-Khatib[d] and Mohammed Anbar[e]

[a]*Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Amman, 19328, Jordan*
[b]*Department of Data Science and Artificial Intelligence, Al-Ahliyya Amman University, Amman, 19328, Jordan*
[c]*Department of Software Engineering, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan*
[d]*Department of Computer Science, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan*
[e]*National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia, Penang, Malaysia*

C H R O N I C L E

Article history:
Received: July 16, 2023
Received in revised format: August 14, 2023
Accepted: September 14, 2023
Available online: September 14, 2023

Keywords:
Machine learning
K-Nearest Neighbors
Malware detection
Distance metric
Cyber-threats

A B S T R A C T

The security and integrity of computer systems and networks highly depend on malware detection. In the realm of malware detection, the K-Nearest Neighbors (KNN) algorithm is a well-liked and successful machine learning algorithm. However, the choice of an acceptable distance metric parameter has a significant impact on the KNN algorithm's performance. This study tries to improve malware detection by adjusting the KNN algorithm's distance metric parameter. The distance metric greatly influences the similarity or dissimilarity between instances in the feature space. The KNN algorithm for malware detection can be more accurate and effective by carefully choosing or modifying the distance metric. This paper analyzes multiple distance metrics, including Minkowski distance, Manhattan distance, and Euclidean distance. These metrics account for the traits of malware samples while capturing various aspects of similarity. The effectiveness of the KNN algorithm is evaluated using the MalMem-2022 malware dataset, and the results are broken down into these three-distance metrics. The experimental findings show that, among the three distance metric parameters, the Euclidean and Minkowski distance metric parameters considerably produced the best outcomes with binary classification. While with multiclass classification, the KNN algorithm has achieved the highest outcomes using Manhattan distance.

1. Introduction

Cyberthreats refer to the possible dangers and attacks that could be launched against computer systems, networks, and other components of the digital infrastructure. The weaknesses in the technology are exploited by these threats to obtain illegal access, disrupt operations, steal data, or inflict damage in various other ways (Lei et al., 2022). DDoS attacks, social engineering, phishing, and malicious software (malware) are typical cyberthreats (Alves, Das, & Morris, 2018; Jain, Rajvaidya, Sah, & Kannan, 2022). Malware is any software or code created with the sole intention of causing damage to computer systems, networks, or devices, exploiting them, or gaining unauthorized access to them. Hackers, often known as cybercriminals, are the individuals or organizations responsible for the harmful creation of this content. Malware can infect various platforms, such as personal computers, server computers, mobile devices, and embedded systems. Malware's principal function is to undermine users' confidence in their computer systems' safety and reliability (Jain, Rajvaidya, Sah, & Kannan, 2022; Peng, Li, Zou, & Wu, 2013; Sen, Aydogan, & Aysan, 2018). According to the 2023 Cyber Threat Report published by SonicWall, malware had its first uptick since 2018, when the number of attacks surged to 5.5 billion, representing a 2% year-over-year increase (Sonicwall, 2022). As of the year 2023, there are 300,000 new cases of malware created each day, 92% of

* Corresponding author.
E-mail address: a.alsaaidah@ammanu.edu.jo (M. M. Abualhaj)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2023.9.012

which are disseminated through email, and it takes an average of 49 days to identify them (Kolesnikov, 2023). Malware can be distributed in various ways, such as through infected websites, malicious email attachments, software downloads from unknown sources, or by exploiting software or operating system vulnerabilities (Qbeitah & Aldwairi, 2018; Das & Nanda, 2013). It is essential for individuals, businesses, and organizations to keep strong cybersecurity measures in place. These steps can include the installation of network firewalls and intrusion detection systems, regularly patching systems and applications, and utilizing well-known antivirus software (Tirumala, Valluri, & Babu, 2019; Belaoued et al., 2020). However, due to the ever-changing nature of the threat landscape, it is imperative to develop new security solutions that are both intelligent and adaptable (Saurabh, 2018; Sai, Tyagi, Panda, & Kumar, 2022). The employment of machine learning (ML) algorithms is an option for accomplishing this goal (Sai et al., 2022; Abualhaj et al., 2022) Malware identification has benefited significantly from applying ML as a tool. The ability of traditional antivirus systems that rely on known patterns or signatures of previously recognized malware to detect new and developing strains of malware is restricted. ML algorithms, on the other hand, have the potential to detect and classify malware based on patterns and features that are automatically learned from massive volumes of data. This presents a significant advantage over traditional algorithms of malware detection. Decision trees (DT), gradient boosting (GB), random forests (RF), support vector machines (SVM), logistic regression (LR), Naive Bayes (NB), and k-nearest neighbors (KNN) are examples of some of the most common ML algorithms (Kolhar et al., 2020; Rosmansyah & Dabarsyah, 2015; Yeo et al., 2018; Choudhary & Sharma, 2020; Chen et al., 2020; Al Zaabi & Mouheb, 2020). KNN is a non-parametric algorithm that assigns a classification to a data point by looking at the class labels of the points in the training data that are closest to it. The effectiveness of KNN is profoundly influenced by the distance metric used (Chen et al., 2020; Hegedus et al., 2011; Alsharaiah et al., 2023). In this study, we will investigate the effectiveness of the KNN algorithm for detecting malware using a variety of distance metrics.

This paper is organized as follows: Section 2 presents some related works. Section 3 discusses the CIC-MalMem-2022 dataset. Section 4 presents the preprocessing of the CIC-MalMem-2022 dataset. Section 5 discusses the KNN algorithm. Section 6 discusses the K-Fold cross-validation technique. Section 7 presents the implementation environment and the results. Section 8 presents the conclusion.

2. Literature review

Rosmansyah and Dabarsyah (2015) has developed an innovative method for detecting malware on Android devices using API classes. The method, detailed in their research paper, leverages ML g techniques to classify programs as benign or malicious. The authors also evaluate the precision rates achieved by implementing ML. The classification process in the study involves using cross-validation and percentage split tests, along with incorporating 51 API package classes derived from 16 API classes. Three classification algorithms—RF, J48, and SVM—are employed to categorize the malware and benign samples. The study encompasses 412 application samples, comprising 205 clean applications and 207 applications containing malicious software. The study results reveal that the RF algorithm achieves an impressive precision rate of 92.4% in the cross-validation test. In comparison, the SVM method attains a precision rate of 91.4% in the percentage split test. Yeo et al. (2018) have automated malware detection using convolutional neural network (CNN) and other ML methods. Traditional malware detection techniques are vulnerable to malware because they have relied on particular packet fields like port numbers and protocols. Instead of depending on port numbers and protocols, the suggested method improves robustness and accuracy by using 35 unique properties retrieved from packet flows. The researchers used information from the Stratosphere IPS project to assess the method's efficacy. This dataset contained normal state packets from an uninfected environment and nine public malware packets. The 35 features were collected from the flows after the packets were transformed into flow data using Netmate. SVM, CNN, multi-layer perceptron (MLP), and RF algorithms were used for classification. The outcomes showed the effectiveness of the suggested method, with both CNN and RF achieving above 85% accuracy, precision, and recall for all classes. Compared to traditional methods that rely on port numbers and protocols, this method presents a more trustworthy and accurate way of identifying malware by utilizing the CNN method and the 35 pertinent features retrieved from packet flows.

Choudhary and Sharma (2020) discuss polymorphic malware, which continuously modifies its identifiable characteristics to evade detection by traditional signature-based systems. The authors employ static information extracted from malicious and non-malicious executable files in a machine-learning algorithm to enable rapid malware prediction. Each algorithm yields distinct classification results. The DT algorithm achieves an accuracy of approximately 99.14%, while the RF algorithm demonstrates an accuracy of around 99.47%. Notably, the Light GBM algorithm surpasses the RF algorithm, achieving an accuracy of nearly 99.50% and attaining the highest level of accuracy. The Light GBM algorithm exhibits efficiency in terms of training time and accuracy, outperforming the RF classifier. Moreover, the Light GBM method offers several advantages, including a significantly reduced false negative rate compared to other classifiers. Chen et al. (2020) develop a model that combines the SVM algorithm and the active learning by learning (ALBL) method to overcome the problem of little labeled data and lessen classification mistakes. The model is made flexible in response to expert feedback, allowing for modifications and advancements over time. Notably, the model gives higher priority to samples with greater levels of uncertainty, offering insightful information about the sources and effects of the data. The researchers used the malware family dataset from the Microsoft Malware Classification Challenge (BIG 2015) on Kaggle to assess the proposed model. The main element was the ALBL method, which enabled iterative model training with consultant input. The model's outstanding accuracy rate of 99.64% was attained. The outcomes unequivocally show that the ALBL method efficiently improves the ML model's functionality

and raises the standard of labeled samples. The model consequently shows a lower classification error rate and quickly reaches its accuracy rate. Al Zaabi and Mouheb (2020) present a machine-learning approach for Android malware detection that utilizes various static features derived from Android Application Packages (APKs). Features, including opcodes, permissions, API calls, activities, and services are highlighted in the study. The researchers gathered the dataset from various raw APK file-providing sources, and AndroPyTool was used to extract dynamic and static information from these files. A total of 4000 APK files were employed in the initial experiment, with 50% categorized as malware and the other 50% as benign programs. Several ML classifiers, including the Linear Classifier, Boosted Trees, Gaussian NB, DT, RF, Gaussian Process (GP), and SVM, are used in the proposed malware detection method. GP produced the highest accuracy of the classifiers (83%), with DT and RF closely behind with 82% accuracy. The GP produced the highest accuracy while taking the longest to train and evaluate the model.

3. CIC-MalMem-2022 Dataset

The Canadian Institute for Cybersecurity developed the CIC-MalMem-2022 to evaluate malware detection algorithms. This paper will utilize the CIC-MalMem-2022 to evaluate the proposed algorithm because it consists of wildly propagated malware in the real world. The CIC-MalMem-2022 consists of three main malware categories, each containing five more malware subcategories, as shown in Table 1. The CIC-MalMem-2022 contains 58,596 records, divided equally between benign and malicious (Dener, Ok, & Orman, 2022). Besides, the CIC-MalMem-2022 contains 55 features.

Table 1
Malware Categories

Malware Main Category	Malware Subcategory	# of Samples
Ransomware	MAZE	1958
	Pysa	1717
	Shade	2128
	Conti	1988
	Ako	2000
Spyware	TIBS	1410
	Gator	2200
	180Solutions	2000
	Transponder	2410
	Coolwebsearch	2000
Trojan Horse	Refroso	2000
	Reconyc	1570
	Zeus	1950
	scar	2000
	Emotet	1967
Total		29,298

4. Data preprocessing

The CIC-MalMem-2022 dataset must be processed to be ready for ML algorithms. The first step of data preprocessing is to convert the textual data into numbers since the KNN algorithm works with numbers (Abualhaj et al., 2022). Only the output (label) field in the CIC-MalMem-2022 dataset contains text, and all other fields contain numbers. As mentioned earlier, the dataset output field contains 15 subcategories of malware plus the benign category (Dener, Ok, & Orman, 2022). The label encoding technique has been used to replace the value of these fields with numbers from 0 to 14 for malware subcategories and 15 for the benign category (Abualhaj et al., 2022). The second step of data preprocessing is ensuring the data has the same scale (Abualhaj et al., 2022). This is done to prevent specific features with large values from dominating over other features with smaller values. To accomplish this goal, the min-max scaling method was applied to scale the values of all features to a range between 0 and 1 (Abualhaj et al., 2022). Table 3 shows a sample of the original CIC-MalMem-2022 dataset before preprocessing. Table 4 shows a sample of the CIC-MalMem-2022 dataset after preprocessing (after applying label encoding and min-max scaling).

Table 3
Sample of the CIC-MalMem-2022 dataset before preprocessing

Features 1 to 9	Label
45, 12, 13.91549774, 0, 313.792842, 2360, 52.44444444, 14120, 313.792842	Benign
39, 15, 11.41025641, 0, 220.5897436, 1562, 40.05128205, 8603, 220.5897436	Ransomware-Ako
38, 15, 9.578947368, 0, 178.3421053, 1057, 35.23333333, 6347, 218.862069	Spyware-Transponder
42, 16, 10.73809524, 0, 209.2142857, 1621, 38.5952381, 8787, 209.2142857	Horse-Emotet

Table 4
Sample of the CIC-MalMem-2022 dataset after preprocessing

Features 1 to 9	Label
0.109589041, 0.0625, 0.808633354, 0, 0.011238179, 0.609448251, 0.984155138, 0.010160989, 0.007197615	0
0.082191781, 0.109375, 0.643469107, 0, 0.007481654, 0.321673278, 0.713782847, 0.004875474, 0.004433016	1
0.077625571, 0.109375, 0.522735517, 0, 0.005778875, 0.139560043, 0.608673288, 0.002714132, 0.004381769	10
0.095890411, 0.125, 0.599155215, 0, 0.00702317, 0.342949874, 0.682017433, 0.005051753, 0.004095596	11

5. KNN algorithm

KNN is one of the most widely used ML algorithms and can be utilized for classification and regression work. It is a non-parametric algorithm, meaning it makes no assumptions about the data distribution used as a basis for its calculations. KNN excels in several areas, including its ease of use and interpretability and its capacity to solve classification issues involving several classes. However, because it needs to calculate distances for every data point, it can be a computationally expensive algorithm, especially when working with huge datasets (Hegedus et al., 2011).

5.1 KNN algorithm processes

The KNN algorithm uses several processes to distinguish malware from benign data. First, as mentioned earlier, the CIC-MalMem-2022 dataset should be loaded and processed to be ready for the KNN algorithm. Then, the value of the K (K-th nearest neighbor) parameters should be defined. In this paper, we will use the values of 3 and 5 for K, as these values have been tested and proved to be the best. After that, determine the distances that separate the newly added data point from the rest of the training data points. Euclidean distance, Manhattan distance, and Minkowski distance are common techniques KNN uses. Selecting the appropriate distance measure significantly impacts how well the KNN algorithm performs, as will be covered in Section 5.2. Next, determine which K neighbors are closest to the newly added data point based on your computed distances. Before the final process, determine the K neighbors' data points that belong to each class and count them. Finally, the class label of the new data point should be assigned based on the class that has the majority among the K neighboring points (Hegedus et al., 2011; Ma & Chi, 2022; Shi et al., 2022). At this stage, the new data point has been determined as malware or benign, and KNN can output the predicted result. Fig. 1 shows the pseudocode of the KNN algorithm.

```
KNN Algorithm
1.  trainingSet: the labeled dataset for training
2.  testInstance: the instance to classify of predict
3.  k: the number of nearest neighbors to consider
4.  for each instance in trainingSet
5.      distance = calculate the distance using Eq. (1), Eq.(2) or Eq. (3)
6.      Distances.append((instance, distance))
7.      Sort(distance) based on distances
8.       Neighbors = []
9.          for i=0 to k-1
10.             neighbors.append(distances[i].instance)
11.             results = majorityVote(neighbors)
12.          end
13. end
```

Fig. 1. Pseudocode of the KNN algorithm

5.2 Distance metric

Choosing the value of the distance measure is one of the key procedures in KNN, as was previously mentioned. The chosen distance metric may impact the KNN algorithm's performance. The nature of the problem and the data will determine which distance measure is best to use, so it's critical to consider these factors. For various types of data, different metrics are appropriate. Euclidean distance, Manhattan distance, and Minkowski distance are popular distance measures (Gao & Li, 2020; Maruf & Laksito, 2020; Zhang, Li, & Li, 2022).

5.2.1. Euclidean distance

Euclidean distance is one of the most common metrics for measuring the distance between two points in a Euclidean space. It is the straight-line distance between two points. In the context of the KNN algorithm, Euclidean distance is often used to measure the distance between data points. When using KNN for classification, the algorithm needs to find the k nearest data points to a given query point and then make predictions based on the labels or values of these neighboring points. Euclidean distance is one way to measure how close each point is to the query point. If you have two points in a 2-dimensional space, P1 (x1, y1) and P2 (x2, y2), the Euclidean distance between these points is calculated using the Pythagorean theorem in Eq. (1) (Gao & Li, 2020; Maruf & Laksito, 2020; Zhang, Li, & Li, 2022).

$$\text{Euclidean Distance} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2)} \tag{1}$$

5.2.2. Manhattan distance

Manhattan distance determines the distance between two points on a grid. Manhattan distance is concerned with the total horizontal and vertical distances between the two points, as opposed to Euclidean distance, which determines the shortest path between two points. When calculating the distance between data points in the KNN method, the Manhattan distance can be

utilized as an alternative to the more popular Euclidean distance. The Manhattan distance can be extremely helpful when working with high-dimensional data or features with diverse scales. For two points in a 2-dimensional space, P1 (x1, y1) and P2 (x2, y2), the Manhattan distance between these points is calculated using Eq. (2) (Gao & Li, 2020; Maruf & Laksito, 2020; Zhang, Li, & Li, 2022).

$$\text{Manhattan Distance} = |x2 - x1| + |y2 - y1| \tag{2}$$

5.2.3. Minkowski distance

Minkowski distance is a generalized metric for determining the separation between two points in an N-dimensional space. Special instances of the Minkowski distance include the Euclidean distance and the Manhattan distance. In a broad sense, Minkowski distance can be used to estimate the separation between data points in the KNN algorithm. For two points P1 and P2 with coordinates ($x_1, x_2, ..., x_n$) and ($y_1, y_2, ..., y_n$), respectively, the Minkowski distance of order p is calculated using Eq. (3) (Gao & Li, 2020; Maruf & Laksito, 2020; Zhang, Li, & Li, 2022).

$$\text{Minkowski Distance} = \left(\sum |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{3}$$

6. K-Fold cross-validation

K-Fold cross-validation is a popular method for evaluating the effectiveness of an ML model. The dataset is divided into five consecutive folds. The model is then trained and tested five times, with a different fold serving as the test set and the remaining folds serving as the training set. The data are then averaged to provide a more precise assessment of the model's performance (Wu et al., 2023). In Fig. 2, the K-Fold cross-validation is shown.

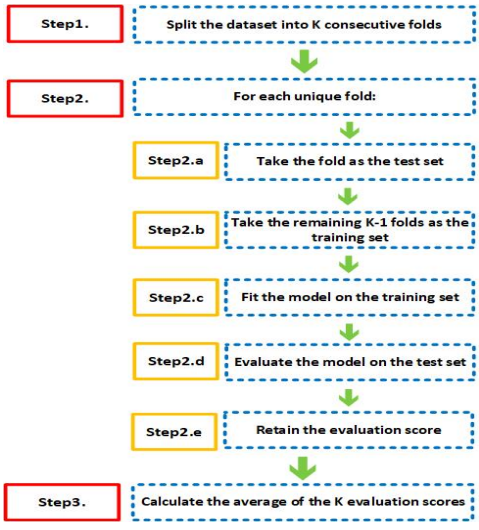

Fig. 2. K-Fold cross-validation operations

6. Implementation and results

For implementation, a laptop with the following specifications is used: Intel Core I7-13620H CPU, SSD 512GB, 16 GB DDR5 RAM, and MS Windows 11. Besides, Python programming, widely used in ML, implements and evaluates the KNN algorithm with different distance metric parameters. Multiclass classification has been used in evaluation experiments. The KNN algorithm has been evaluated based on the elements of the confusion matrix: True positive (TPo), True Negative (TNe), False Positive (FPo), and False Negative (FNe). Using these elements, four metrics were built to evaluate the usefulness of the KNN algorithm to detect malware. Accuracy (Acr) is the first metric. The Acr is determined using Eq. (4). Recall (Rec) is the second metric. The Rec is determined using Eq. (5). Precision (Pre) is the third metric. The Pre is calculated using Eq. (6). F1-Score is the final metric. The F1-Score is calculated using Eq. (7) (Wu et al., 2023; Al-Mimi et al., 2023).

$$Acr = \frac{(TPo + TNe)}{(TPo + TNe + FPo + FNe)} \tag{4}$$

$$Rec = \frac{TPo}{(TPo + FNe)} \tag{5}$$

$$Pre = \frac{TPo}{(TPo + FPo)} \tag{6}$$

$$F1 - score = 2 \times \frac{Pre \times Rec}{Pre + Rec} \quad (7)$$

As mentioned earlier, the choice of distance metric in the KNN algorithm is impacting the KNN algorithm performance. For that, different values of distance metrics are tested: the Euclidean distance, Manhattan distance, and Minkowski distance. Fig. 3 (a), 3 (b), and 3 (c) display the Acr of the KNN algorithm with binary classification, multiclass classification (4 classes), and multiclass classification (16 classes), respectively. With binary classification, the Acr of the KNN has attained the highest values of 99.974% when k is 3 for both Euclidean distance and Minkowski distance. However, multiclass classification (4 classes) and multiclass classification (16 classes) of the KNN have attained the highest values of 82.218% and 66.937%, respectively, when k is 3 with Manhattan distance.

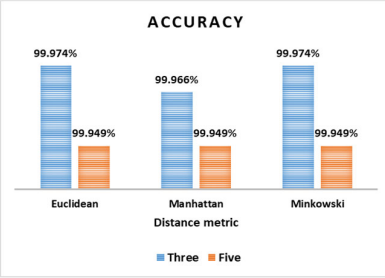

Fig. 3. (a) Accuracy of binary classification

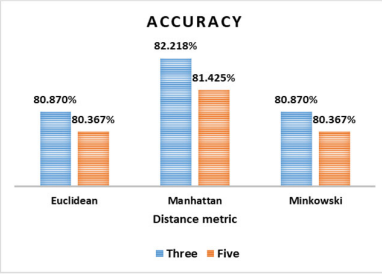

Fig. 3. (b) Accuracy of multiclass classification (4 classes)

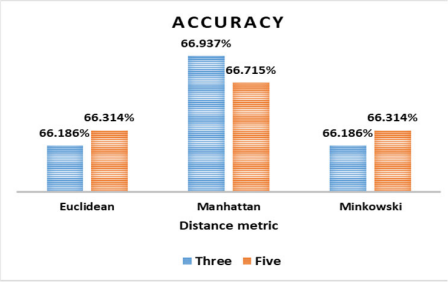

Fig. 3. (c) Accuracy of multiclass classification (16 classes)

The KNN has attained the same exact value of Acr as Rec, as shown in Fig. 4 (a), 4 (b), and 4 (c). Fig. 5 (a), 5 (b), and 5 (c) display the Pre of the KNN algorithm with binary classification, multiclass classification (4 classes), and multiclass classification (16 classes), respectively. Fig. 6 (a), 6 (b), and 6 (c) display the F1-score of the KNN algorithm with binary classification, multiclass classification (4 classes), and multiclass classification (16 classes), respectively. With binary classification, the pre and F1-score of the KNN have attained the same highest values (99.974%) of Acr and Rec when k is 3 for both Euclidean distance and Minkowski distance. However, multiclass classification (4 classes) and multiclass classification (16 classes) of KNN have attained the highest Rec values (82.397% and 66.888%, respectively) and the highest F1-score values (82.247% and 67.329%, respectively), when k is 3 with Manhattan distance.

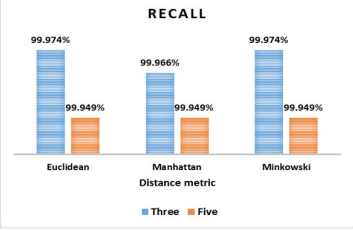

Fig. 4. (a) Recall of binary classification

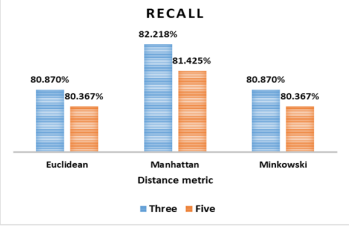

Fig. 4. (b) Recall of multiclass classification (4 classes)

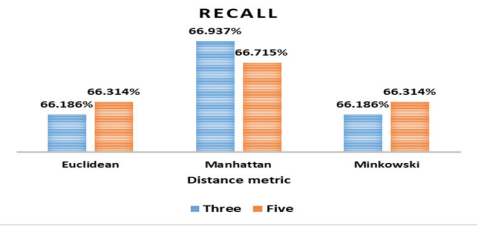

Fig. 4. (c) Recall of multiclass classification (16 classes)

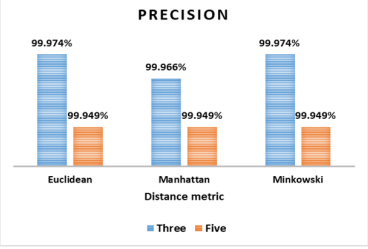

Fig. 5. (a) Precision of binary classification

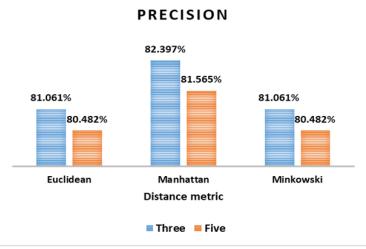

Fig. 5. (b) Precision of multiclass classification (4 classes)

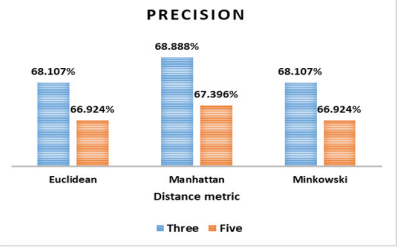

Fig. 5. (c) Precision of multiclass classification (16 classes)

In summary, the KNN algorithm has achieved the highest Acr, Rec, Pre, and F1-score with the three classification types when k is 3. On the other hand, the KNN algorithm has achieved the highest Acr, Rec, Pre, and F1-score with binary classification using Euclidean distance and Minkowski distance. While with multiclass classification (4 classes) and multiclass classification (16 classes), the KNN algorithm has achieved the highest Acr, Rec, Pre and, F1-score using Manhattan distance.

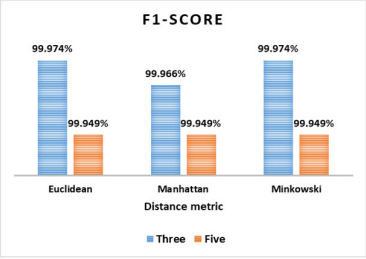

Fig. 6. (a) F1-score of binary classification

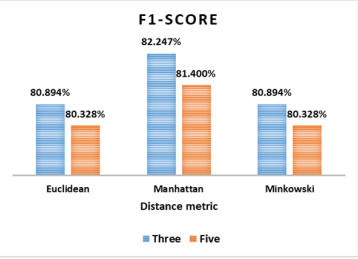

Fig. 6. (b) F1-score of multiclass classification (4 classes)

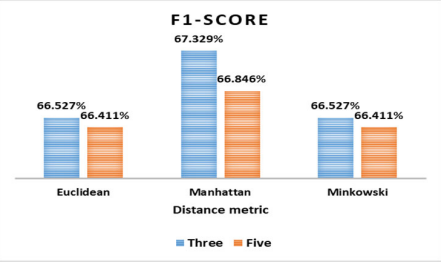

Fig. 6. (c) F1-score of multiclass classification (16 classes)

7. Conclusion

In conclusion, this research aimed to improve the identification of malware by adjusting a parameter in the KNN algorithm that was related to distance metrics. It is possible to dramatically improve the performance of the algorithm by picking an appropriate distance measure, which will ultimately result in the identification of malware samples that is both more accurate and dependable. When several distance metrics, like the Euclidean distance, the Manhattan distance, and the Minkowski distance, were looked at, it became clear that different distance metrics show different aspects of how similar the different instances of the feature space are. To conduct the study, the MalMem-2022 malware dataset was utilized. It was discovered that, out of the three different distance metrics considered, the Euclidean distance and Minkowski distance consistently produced the best results with binary classification. In contrast, Manhattan distance produced the best results with multiclass classification. This study's results contribute to the growing field of malware detection by illuminating the influence that distance metrics have on how well the KNN algorithm does its job. Security analysts and practitioners can use the optimized parameter settings to improve the accuracy and reliability of malware detection systems, enhancing the overall security posture of computer networks and systems. This can be done by using the capabilities of the optimized parameter settings.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

For this research work all authors' have equally contributed in related works, method design, implementation, and writing.

References

Abualhaj, M. M., Abu-Shareha, A. A., Hiari, M. O., Alrabanah, Y., Al-Zyoud, M., & Alsharaiah, M. A. (2022). A Paradigm for DoS Attack Disclosure using Machine Learning Techniques. *International Journal of Advanced Computer Science and Applications*, *13*(3).

Al Zaabi, A., & Mouheb, D. (2020, November). Android malware detection using static features and machine learning. In *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)* (pp. 1-5). IEEE.

Al-Mimi, H., Hamad, N. A., Abualhaj, M. M., Daoud, M. S., Al-dahoud, A., & Rasmi, M. (2023). An Enhanced Intrusion Detection System for Protecting HTTP Services from Attacks. *International Journal of Advances in Soft Computing & Its Applications*, *15*(2).

Alsharaiah, M., Abu-Shareha, A., Abualhaj, M., Baniata, L., Adwan, O., Al-saaidah, A., & Oraiqat, M. (2023). A new phishing-website detection framework using ensemble classification and clustering. *International Journal of Data and Network Science, 7*(2), 857-864.

Alves, T., Das, R., & Morris, T. (2018). Embedding encryption and machine learning intrusion prevention systems on programmable logic controllers. *IEEE Embedded Systems Letters*, *10*(3), 99-102.

Belaoued, M., Derhab, A., Mazouzi, S., & Khan, F. A. (2020). MACoMal: A multi-agent based collaborative mechanism for anti-malware assistance. *IEEE Access*, *8*, 14329-14343.

Chen, C. W., Su, C. H., Lee, K. W., & Bair, P. H. (2020, February). Malware family classification using active learning by learning. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)* (pp. 590-595). IEEE.

Choudhary, S., & Sharma, A. (2020, February). Malware detection & classification using machine learning. In *2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3)* (pp. 1-4). IEEE.

Das, D., & Nanda, S. (2013, December). Securing computer networks by networking multiple OS kernels. Revisting network security: protecting computer networks from malwares. In *World Congress on Internet Security (WorldCIS-2013)* (pp. 95-98). IEEE.

Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in big data environment. *Applied Sciences*, *12*(17), 8604.

Gao, X., & Li, G. (2020). A KNN model based on manhattan distance to identify the SNARE proteins. *Ieee Access*, *8*, 112922-112931.

Hegedus, J., Miche, Y., Ilin, A., & Lendasse, A. (2011, December). Methodology for behavioral-based malware analysis and detection using random projections and k-nearest neighbors classifiers. In *2011 seventh international conference on computational intelligence and security* (pp. 1016-1023). IEEE.

Jain, P., Rajvaidya, I., Sah, K. K., & Kannan, J. (2022, February). Machine Learning Techniques for Malware Detection-a Research Review. In *2022 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (pp. 1-6). IEEE.

Kolesnikov, N. (2023). 50+ cybersecurity statistics for 2023 you need to know – where, who & what is targeted. Techopedia. https://www.techopedia.com/cybersecurity-statistics.

Kolhar, M., Al-Turjman, F., Alameen, A., & Abualhaj, M. M. (2020). A three layered decentralized IoT biometric architecture for city lockdown during COVID-19 outbreak. *Ieee Access*, *8*, 163608-163617.

Lei, J., Gao, S., Shi, J., Wei, X., Dong, M., Wang, W., & Han, Z. (2022). A Reinforcement Learning Approach for Defending Against Multiscenario Load Redistribution Attacks. *IEEE Transactions on Smart Grid*, *13*(5), 3711-3722.

Ma, C., & Chi, Y. (2022). KNN normalized optimization and platform tuning based on hadoop. *IEEE Access*, *10*, 81406-81433.

Maruf, Z. R., & Laksito, A. D. (2020, November). The comparison of distance measurement for optimizing KNN collaborative filtering recommender system. In *2020 3rd International Conference on Information and Communications Technology (ICOIACT)* (pp. 89-93). IEEE.

Peng, W., Li, F., Zou, X., & Wu, J. (2013). Behavioral malware detection in delay tolerant networks. *IEEE Transactions on Parallel and Distributed systems*, *25*(1), 53-63.

Qbeitah, M. A., & Aldwairi, M. (2018, April). Dynamic malware analysis of phishing emails. In *2018 9th International Conference on Information and Communication Systems (ICICS)* (pp. 18-24). IEEE.

Rosmansyah, Y., & Dabarsyah, B. (2015, August). Malware detection on android smartphones using API class and machine learning. In *2015 International Conference on Electrical Engineering and Informatics (ICEEI)* (pp. 294-297). IEEE.

Sai, M., Tyagi, A., Panda, K., & Kumar, S. (2022, November). Machine learning-based malware detection using stacking of opcodes and bytecode sequences. In *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 204-209). IEEE.

Saurabh. (2018, December). Advance malware analysis using static and dynamic methodology. In *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)* (pp. 1-5). IEEE.

Sen, S., Aydogan, E., & Aysan, A. I. (2018). Coevolution of mobile malware and anti-malware. *IEEE Transactions on Information Forensics and Security*, *13*(10), 2563-2574.

Shi, K., Chen, S., Li, D., Tian, K., & Feng, M. (2022, November). Analysis of the Optimized KNN Algorithm for the Data Security of DR Service. In *2022 IEEE 6th Conference on Energy Internet and Energy System Integration (EI2)* (pp. 1634-1637). IEEE.

Sonicwall, 2022 sonicwall cyber threat report. https://www.infopoint-security.de/media/2022-sonicwall-cyber-threat-report.pdf

Tirumala, S. S., Valluri, M. R., & Babu, G. A. (2019, January). A survey on cybersecurity awareness concerns, practices and conceptual measures. In *2019 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-6). IEEE.

Wu, H., Han, M., Chen, Z., Li, M., & Zhang, X. (2023). A Weighted Ensemble Classification Algorithm Based on Nearest Neighbors for Multi-Label Data Stream. *ACM Transactions on Knowledge Discovery from Data*, *17*(5), 1-21.

Yeo, M., Koo, Y., Yoon, Y., Hwang, T., Ryu, J., Song, J., & Park, C. (2018, January). Flow-based malware detection using convolutional neural network. In *2018 International Conference on Information Networking (ICOIN)* (pp. 910-913). IEEE.

Zhang, S., Li, J., & Li, Y. (2022). Reachable distance function for KNN classification. *IEEE Transactions on Knowledge and Data Engineering*.