

# Tuning the K value in K-nearest neighbors for malware detection

Mosleh M. Abualhaj<sup>1</sup>, Ahmad Adel Abu-Shareha<sup>2</sup>, Qusai Y. Shambour<sup>3</sup>,  
Sumaya N. Al-Khatib<sup>1</sup>, Mohammad O. Hiari<sup>1</sup>

<sup>1</sup>Department of Networks and Cybersecurity, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan

<sup>2</sup>Department of Data Science and Artificial Intelligence, Faculty of Information Technology,  
Al-Ahliyya Amman University, Amman, Jordan

<sup>3</sup>Department of Software Engineering, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan

## Article Info

### Article history:

Received Jul 24, 2023

Revised Oct 29, 2023

Accepted Nov 15, 2023

### Keywords:

Cybersecurity  
K parameter tuning  
K-nearest neighbors  
Machine learning  
Malware detection

## ABSTRACT

Malicious software, also referred to as malware, poses a serious threat to computer networks, user privacy, and user systems. Effective cybersecurity depends on the correct detection and classification of malware. In order to improve its effectiveness, the K-nearest neighbors (KNN) method is applied systematically in this study to the task of malware detection. The study investigates the effect of the number of neighbors (K) parameter on the KNN's performance. MalMem-2022 malware datasets and relevant evaluation criteria like accuracy, precision, recall, and F1-score will be used to assess the efficacy of the suggested technique. The experiments evaluate how parameter tuning affects the accuracy of malware detection by comparing the performance of various parameter setups. The study findings show that careful parameter adjustment considerably boosts the KNN method's malware detection capability. The research also highlights the potential of KNN with parameter adjustment as a useful tool for malware detection in real-world circumstances, allowing for prompt and precise identification of malware.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Mosleh M. Abualhaj  
Department of Networks and Cybersecurity, Faculty of Information Technology  
Al-Ahliyya Amman University  
Amman, 19328, Jordan  
Email: m.abualhaj@ammanu.edu.jo

## 1. INTRODUCTION

Businesses and individuals now depend more on technology and are more networked, which presents a variety of cyber risks. Cyber risks are potential dangers and weaknesses in digital systems, networks, and information that could result in unauthorized access, data breaches, monetary loss, reputational harm, or business interruption [1]. Weak or stolen passwords, phishing, DoS attacks, and malicious software (Malware) are a few typical instances of cyber risks [2], [3]. Any software intended to damage, exploit, or allow illegal access to computer systems, networks, or devices is referred to as malware. Malware comes in a variety of forms, each with unique traits and attack strategies. Viruses, Worms, Trojan horses, and Ransomware are examples of common types of malware [3], [4]. The total number of malware attacks in 2022 was 5.5 billion [5]. This inevitably results in significant data breaches, losses, or corruption.

It is crucial to adopt security measures such as updating operating systems and apps with security patches, training users, and deploying reliable anti-malware software in order to safeguard against malware [6], [7]. On the other hand, advanced malware is a cunning and elusive strategy that hackers use to circumvent traditional security measures and commit crimes. These cutting-edge malware techniques demonstrate how

constantly changing cyber risks are and how urgent it is for businesses to use cutting-edge security measures that go beyond typical anti-malware solutions [8], [9]. Machine learning (ML) approaches can be used to protect against malware by improving the detection and prevention capabilities of security systems [9], [10]. The goal of ML is to create methods and models that let computers learn and make predictions or judgments without having to be explicitly programmed. It entails building and training mathematical models on data, which the models can then use to predict the future, spot patterns, or obtain new knowledge. Malware detection, behavioral analysis, dynamic analysis, and feature extraction are just a few of the numerous ways that ML can be used in the context of malware. ML methods can be broadly divided into three categories: reinforcement learning, unsupervised learning, and supervised learning (SL) [9], [10].

SL is an ML technique where a model is trained on a labeled dataset that pairs input data with corresponding target labels or outcomes. To correctly forecast or categorize new, unexplored data, the model must learn the link between the input features and the target variable. Because it can identify patterns and traits of malware from labeled data, SL is a widely utilized approach in malware detection. The labeled dataset's quality, the feature selection, and the choice of the best methods all affect how well malware may be detected using SL [10]–[12]. The data's type, the virus's complexity, and the required trade-offs between accuracy, performance, and interpretability all play a role in choosing the best methods for malware detection. In the area of malware detection, decision trees (DT), random forests (RF), support vector machines (SVM), naive bayes (NB), and K-nearest neighbors (KNN) are some of the commonly employed methods. KNN is a straightforward and understandable method that classifies data points based on the consensus of those points in the feature space that are closest to them [13]–[17]. It is appropriate for scenarios in which neighborhood and local patterns play a significant role in malware identification. In this study, an ML model called malware- KNN (MW-KNN) that uses an optimized KNN algorithm will be built for malware detection.

## 2. RELATED WORKS

Goyal and Kumar [13] discuss the difficulty of identifying malware, particularly in light of the volume of malware that is produced and spread on a daily basis. The major goal of the study is to minimize harm through early malware detection. The pipeline procedure for both signature-based and behavior-based malware detection algorithms is thoroughly explained by Goyal and Kumar [13]. Using a dataset of 1494 malware and 1347 benign samples, the authors ran an experiment. These samples were used to extract two different types of features: non-repetitive consecutive application programming interface (API) calls for dynamic analysis and string features for static analysis. After that, they used different ML methods on these attributes with training/testing ratios of 80:20, 70:30, and 60:40. Gaussian NB, multi NB, DT, RF, KNN, and SVM are the ML methods. The findings indicated that dynamic features are more promising than static features, as the accuracy with the API calls feature was higher than the accuracy with the string feature. With an accuracy of 97.53%, the RF algorithm on API calls produced the best results. The behavior-based approach is more promising, according to the authors, for identifying new malware.

Malware program classification is a challenge that Davuluru *et al.* [14] address. The purpose of this paper is to investigate the performance of different convolutional neural network (CNN)-based architectures (AlexNet, ResNet, and very deep convolutional [VGG16]) as feature extractors and classification tools after the visualization of malware programs. The authors suggest a fusion strategy that combines CNN, which has been producing cutting-edge results for image-based classification, with the pattern recognition approach, which proved successful for classifying malware. They use classic ML methods like SVM and KNN to classify by extracting features from the suggested CNN architectures. For a set of 2,174 test samples taken from the BIG 2015 dataset, the suggested algorithm achieves an overall accuracy of 99.4%. The findings unambiguously show that CNN is useful for categorizing malware programs as a feature extractor as well as a classification tool. The performance of algorithms is studied to aid subject-matter experts in selecting the best algorithm for their purposes.

According to Narayanan *et al.* [15], existing malicious groups and classes are polymorphic, which makes it challenging for conventional malware detection techniques to work properly. By visualizing viruses in an image format that captures minute changes while preserving a global framework, the study aims to improve malware categorization. As a result, it will be clear that malware classification can be enhanced when approached as an image classification issue. The principal component analysis (PCA) is implemented for feature extraction. The performance of various artificial neural network (ANN) algorithms, along with KNN and SVN methods, is studied for the identification of malware data into their respective classes. The findings imply that each malware program in a family has a unique pattern. These patterns are easily distinct between families and are relatively similar within one family. Because picture patterns for malware programs from the same family tend to be similar, the authors discovered that the KNN classifier performs well. The outcomes also show that PCA transformation is the best option in this case.

Hegedus *et al.* [16] discuss malware detection and pay particular attention to the drawbacks of signature-based malware detection. Because of the growth of polymorphic and metamorphic malware, the authors emphasize the necessity of execution-level identification. The study's goal is to enhance malware detection by offering a two-stage process that makes use of the KNN method's random projections. According to the study, a set of samples is first pruned, and only the samples that satisfy a particular requirement are kept. They are then regarded as "unpredictable" and perhaps "clean" samples. To find potential false negatives, the authors next employ the KNN method and Jaccard similarity measure. They also go through how the confusion matrix is affected by the random projection dimension and how to leverage it to get better outcomes. The results demonstrate that raising the projected vectors' dimensions improves the outcomes: the proportion of unpredictable samples declines, while the true positives rise and the false positives fall. The authors also point out that when dimensions increase, the true and false negatives seldom alter.

Şahn *et al.* [17] draw attention to the rising incidence of malware on Android devices. The study's goal is to make Android malware detection better by advocating a permission weight strategy. The authors provide a weighting mechanism that uses the KNN and NB methods for malware identification and gives each permission a unique score. The relevance frequency method, a successful weighting method in text categorization, is also covered by the authors as it relates to Android malware detection. The outcomes demonstrate that the suggested strategy produces superior outcomes compared to earlier ones. Both accuracy and F-score showed an average improvement of 2% with the KNN method. The accuracy and F-score metrics revealed an average improvement of 4% and 7% with the NB method, respectively. When comparing the classification methods, the KNN method produced the greatest results for the accuracy metric, whereas Gaussian NB produced the best results for the F-score metric.

### 3. CIC-MALMEM-2022 DATASET

A malware dataset called CIC-MalMem-2022 is used to evaluate malware detection techniques in this study. Using malware that is common in the real world, the dataset was developed to replicate a situation as closely as possible to the real world. The dataset is balanced with 50% malicious memory dumps and 50% benign memory dumps. There are 58,596 records total in the dataset, 29,298 of which are benign and 29,298 of which are malicious. The dataset includes the three primary malware categories of Trojan Horse, Ransomware, and Spyware. There are five subcategories within each of these categories. The subcategories of Trojan Horse are Zeus (1,950 samples), Emotet (1,967 samples), Refroso (2,000 samples), scar (2,000 samples), and Reconyc (1,570 samples). The subcategories of Ransomware are Conti (1,988 samples), MAZE (1,958 samples), Pysa (1,717 samples), Ako (2,000 samples), and Shade (2,128 samples). The subcategories of Spyware are 180Solutions (2,000 samples), Coolwebsearch (2,000 samples), Gator (2,200 samples), Transponder (2,410 samples), and TIBS (1,410 samples). The dataset also includes 55 attributes that were utilized to differentiate the various malware groups [18].

### 4. METHOD

This section discusses the proposed model that will be used to detect the Malware. This model includes preparing the data for the classification algorithm and the used KNN algorithm. Two main steps will be performed to prepare the data: transformation and normalization, as discussed in sections 4.1 and 4.2, respectively. Section 4.3 discusses the operation and parameter values of the KNN classifiers.

#### 4.1. Transformation

The term "transformation" refers to changing non-numerical data's value to a numerical one. Since most ML methods work with numerical data, it is frequently required to convert data into numerical representations. One of the most common methods of data transformation is label encoding. It gives each category in the variable a special numerical value. When the category variable has an intrinsic order or ranking, this encoding is often utilized [10]. Regarding the MalMem-2022 that is being utilized, the output field is textual and contains 4 values of main categories and 16 values of sub categories [18]. The label encoding approach was used to convert these values into numbers. The main categories are numbered 0, 1, 2, and 3. At the same time, the sub categories are numbered by 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15.

#### 4.2. Normalization

Normalization is used in ML to scale numerical features to a standard range or distribution. It seeks to level the scale between all features, preventing one feature from monopolizing learning due to its greater magnitude. The popular data normalizing method known as "min-max scaling" is used in ML to scale numerical features to a particular range, usually between 0 and 1. It maintains the relative ordering of the data points while linearly transforming the original values to a normalized scale [10]. The min-max scaling formula

is shown in (1). Where  $val$  is the original value of the feature,  $n\_val$  is the new value after normalization,  $mi\_val$  is the minimum value in the feature, and  $ma\_val$  is the maximum value in the feature.

$$n\_val = \frac{(val - mi\_val)}{(ma\_val - mi\_val)} \quad (1)$$

### 4.3. KNN classification algorithm

KNN is a straightforward and popular classification method. It is a form of instance-based learning, or lazy learning, where all computation is postponed until after the function has been evaluated and the function is only locally approximated. It is a simple algorithm that is easy to comprehend. It doesn't use any mathematical models or make any assumptions about the distribution of the underlying data. Additionally, because KNN makes no assumptions about the distribution of the data, it is fairly resistant to outliers. Furthermore, KNN offers interpretability by making predictions understandable and explicable by looking at nearby data points and their corresponding class labels [10], [13]–[17]. The KNN is a good option for the suggested MW-KNN model to identify malware because of all these qualities.

#### 4.3.1. KNN operations

During the prediction phase, the KNN classifier performs a number of operations. First, choose the value of  $K$ . Establish the  $K$ -th nearest neighbor to be taken into account for classification. As will be covered in Section 4.3.2, the value of  $k$  has a significant impact on how well the KNN method performs. Second, the distance between each sample in the training set and the input sample should be calculated. Euclidean distance is the most common distance metrics used in KNN. Third, determine the majority class. Count each class label's appearances among the  $K$  closest neighbors, and then designate the class label that does so most frequently as the predicted class for the input sample. Fourth, give the input sample the predicted class label. Figure 1 summarizes these operations. The value of  $k$  (number of neighbors) need to be adjusted, as was already mentioned. The parameter has a significant impact on how well the model performs [10], [13]–[17]. Sections 4.3.2 covers how to choose the value of  $k$ .

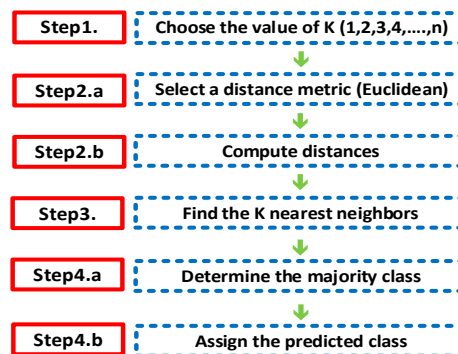


Figure 1. KNN method operations

#### 4.3.2. The value of $k$

As was already said, selecting the value of  $k$  is one of the key operations in KNN. It's critical to understand that there is no one ideal value for  $k$ . The best value for  $k$  should be determined through methodical testing and validation and depends on the particular dataset and situation at hand. The system's performance will be enhanced by selecting the proper value. When deciding on the value of  $k$ , there are numerous rules and factors to take into account. First, choosing a small value for  $k$  makes KNN capture the minute details in the data. It may, however, also catch noise, making it more prone to overfitting. Second, the decision border gets smoother and less elastic as the value of  $k$  increases. This may aid in lowering the model's variance, but at the expense of boosting bias. Underfitting may result from  $k$  values that are extremely large. Third, picking odd numbers for  $k$  is frequently done in order to prevent ties, particularly in binary classification issues. Ties can still happen in multiclass classification issues even with odd values for  $k$ , although going with odd numbers can cut down on the possibility of ties. Fourth, a common rule is to set  $k$  to be equal to the square root of the number of samples in the training dataset [19]–[23]. The suggested model will test up to 10 odd values of  $k$ , counting from 3 and increasing by 4 (for example, 3, 7, and 11). The MalMem-2022 dataset's square root of the number of samples will also be used. The suggested MW-KNN model will use the value that gives the best performance.

## 5. RESULTS AND DISCUSSION

A laptop with the following specifications is used to implement the suggested MW-KNN model: Intel Core I7-13620H CPU, 16 GB DDR5 RAM, Nvidia GeForce RTX 4050 6GB DDR6 graphic card, SSD 512GB M.2 storage, and Windows 11 OS. The MW-KNN model was created using Python programming. Python is widely used in many fields, including machine learning. It offers a robust ecosystem of tools and libraries that make it simple to create and use ML models. The MW-KNN model was constructed using a number of libraries, including scikit-learn and numpy.

The suggested MW-KNN model has been assessed using the four components of the popular confusion matrix. These components are true positive, true negative, false positive, and false negative. On top of these components, five measures were constructed to assess the effectiveness of the MW-KNN model. Accuracy is the first metric. Accuracy is the proportion of accurate forecasts to all predictions. Recall is the second metric. Recall measures how many true positive predictions there were compared to all other True positive and False Positive forecasts. Precision is the third metric. Precision is the proportion of true positive forecasts to all true positive and false positive predictions. F1-Score is the final metric. For unbalanced datasets, the F1-Score, which is the harmonic mean of precision and recall, performs better than accuracy [10], [24], [25].

As mentioned earlier, the choice of k metric in KNN classifier is impacting the proposed MW-KNN model performance. For that, different values of this parameter are tested. The tested values are 3, 5, 7, 11, 15, 19, 23, 27, 31, 34, 37, and 217 (Square root of the training dataset). K-Fold cross-validation is used to divide the dataset into 5 consecutive folds. The model is then trained and tested 5 times, with a different fold serving as the test set and the remaining folds serving as the training set.

Figures 2(a) and 2(b) show the Accuracy of the MW-KNN model with binary classification and multiclass classification, respectively. As we can see, with binary classification, the Accuracy of the model has achieved the highest values of 99.974% when k is equal to 3. However, with multiclass classification, the Accuracy of the model has achieved the highest value of 66.314% when k is equal to 5. Figures 3(a) and 3(b) show the Recall of the MW-KNN model with binary classification and multiclass classification, respectively. As we can see, with binary classification, the Recall of the model has achieved the highest values of 99.974% when k is equal to 3. However, with multiclass classification, the Recall of the model has achieved the highest value of 66.314% when k is equal to 5. Figures 4(a) and 4(b) show the Precision of the MW-KNN model with binary classification and multiclass classification, respectively. As we can see, with the two types of classification, the Precision of the model has achieved the highest values of 99.974% and 68.107%, respectively, when k is equal to 3. Figures 5(a) and 5(b) show the F1-score of the MW-KNN model with binary classification and multiclass classification, respectively. As we can see, with the two types of classification, the F1-score of the model has achieved the highest values of 99.974%, and 66.527%, respectively, when k is equal to 3.

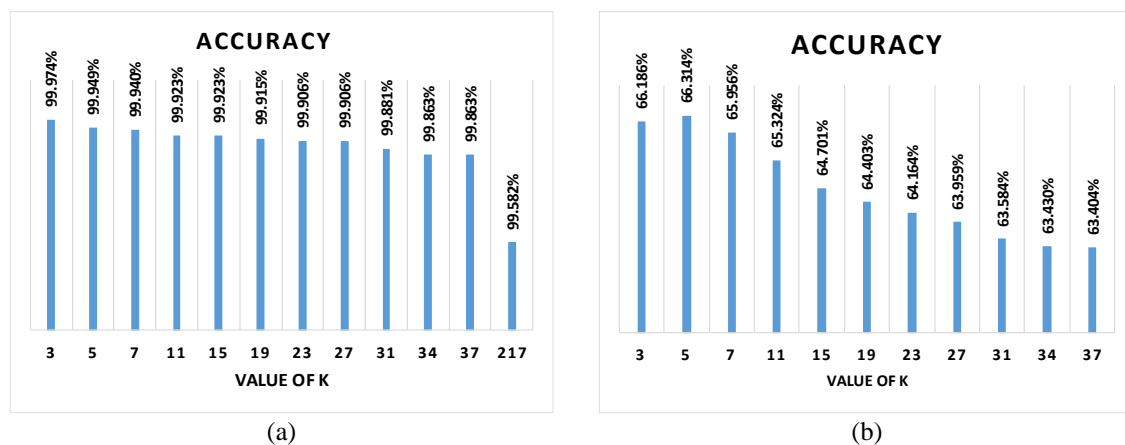


Figure 2. Accuracy of (a) binary classification and (b) multiclass classification

In summary, the MW-KNN model has achieved the highest accuracy and recall with multiclass classification when k is equal to 5. On the other hand, for all other k values of all four metrics (Accuracy, Recall, Precision, and F1-Score) with the two classification types, the MW-KNN model has achieved the highest results when k is equal to 3. Finally, in general, when the value of k is increased, the achievement of the model decreases with all five metrics.

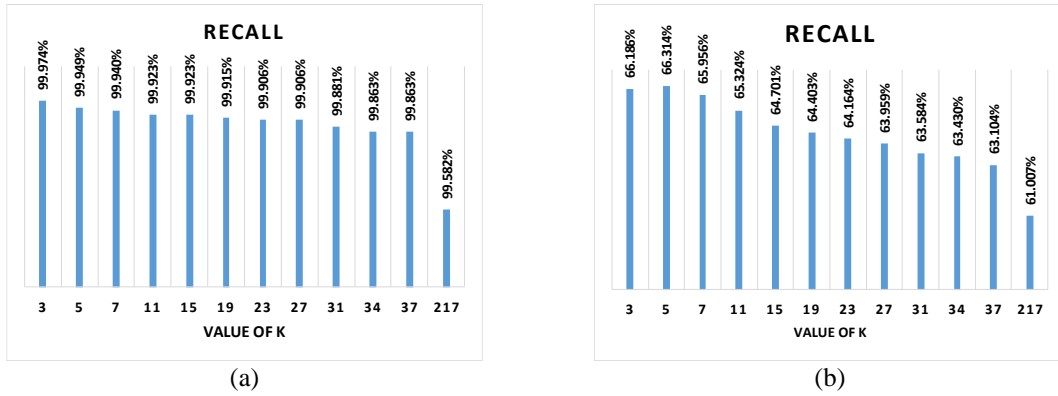


Figure 3. Recall of (a) binary classification and (b) multiclass classification

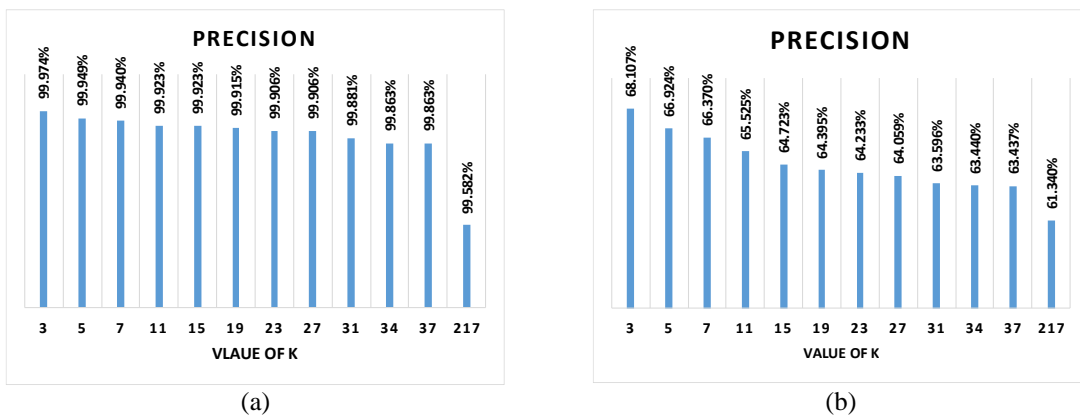


Figure 4. Precision of (a) binary classification and (b) multiclass classification

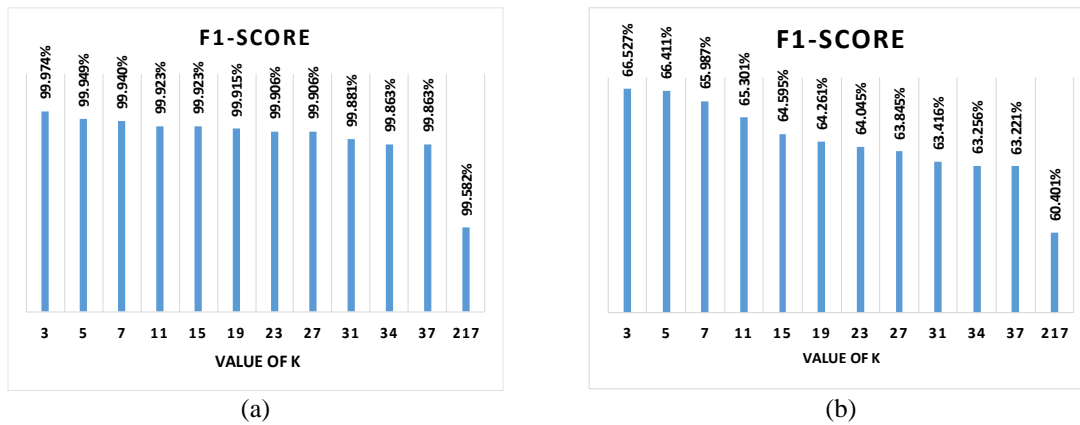


Figure 5. F1-score of (a) binary classification and (b) multiclass classification

## 6. CONCLUSION

This study suggested utilizing the KNN method with parameter adjustment to create the MW-KNN model for malware detection. The model tries to improve the performance and effectiveness of malware detection systems by utilizing the KNN's advantages and tweaking its parameters. We have shown that changing the parameters is a key part of improving the Accuracy, Recall, Precision, MCC, and F1-Score of the KNN algorithm for finding malware. We did this by looking at the number of neighbors (K) in a systematic way. The results highlight the significance of selecting proper parameter values with attention in order to obtain the best outcomes. Tests carried out with datasets from MalMem-2022. We found that parameter tuning considerably increases the accuracy of malware classification by comparing the performance of tuned




parameter values. The MW-KNN model has a lot of potential for cybersecurity since it addresses the urgent need for effective malware detection techniques. The model provides a strong foundation for the precise identification and classification of dangerous software by leveraging the KNN method, which is renowned for its simplicity and efficacy in classification tasks. To fully assess the MW-KNN model's efficacy in real-world circumstances, however, more investigation and testing are required. Thorough testing, benchmarking against current detection systems, and computing efficiency analyses will determine its viability and scalability.

## REFERENCES




- [1] P. Lau, L. Wang, W. Wei, Z. Liu, and C.-W. Ten, "A novel mutual insurance model for hedging against cyber risks in power systems deploying smart technologies," *IEEE Transactions on Power Systems*, vol. 38, no. 1, pp. 630–642, Jan. 2023, doi: 10.1109/TPWRS.2022.3164628.
- [2] O. I. Falowo, S. Popoola, J. Riep, V. A. Adewopo, and J. Koch, "Threat actors' tenacity to disrupt: examination of major cybersecurity incidents," *IEEE Access*, vol. 10, pp. 134038–134051, 2022, doi: 10.1109/ACCESS.2022.3231847.
- [3] D.-O. Won, Y.-N. Jang, and S.-W. Lee, "PlausMal-GAN: Plausible malware training based on generative adversarial networks for analogous zero-day malware detection," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 1, pp. 82–94, Jan. 2023, doi: 10.1109/TETC.2022.3170544.
- [4] K. A. Dhanya et al., "Obfuscated malware detection in IoT android applications using markov images and CNN," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2756–2766, Jun. 2023, doi: 10.1109/JSYST.2023.3238678.
- [5] A. Petrosyan, "Number of malware attacks per year 2022," *Statista*, 2022, [Online]. Available: <https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/>
- [6] Y. Zhang et al., "Looking back! Using early versions of android apps as attack vectors," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 652–666, Mar. 2021, doi: 10.1109/TDSC.2019.2914202.
- [7] M. Belaoued, A. Derhab, S. Mazouzi, and F. A. Khan, "MACoMal: A multi-agent based collaborative mechanism for anti-malware assistance," *IEEE Access*, vol. 8, pp. 14329–14343, 2020, doi: 10.1109/ACCESS.2020.2966321.
- [8] Y. Guo, C.-W. Ten, S. Hu, and W. W. Weaver, "Preventive maintenance for advanced metering infrastructure against malware propagation," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1314–1328, May 2016, doi: 10.1109/TSG.2015.2453342.
- [9] A. Abusnaina et al., "DL-FHMC: Deep learning-based fine-grained hierarchical learning approach for robust malware classification," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3432–3447, Sep. 2022, doi: 10.1109/TDSC.2021.3097296.
- [10] M. M. Abualhaj, A. A. Abu-Shareha, M. O. Hiari, Y. Alrabanah, M. Al-Zyouid, and M. A. Alsharaiah, "A paradigm for DoS attack disclosure using machine learning techniques," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, 2022, doi: 10.14569/IJACSA.2022.0130325.
- [11] S. D. S.L and J. C.D, "Windows malware detector using convolutional neural network based on visualization images," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 1057–1069, Apr. 2021, doi: 10.1109/TETC.2019.2910086.
- [12] M. Kolhar, F. Al-Turjman, A. Alameen, and M. M. Abualhaj, "A three layered decentralized iot biometric architecture for city lockdown durin COVID-19 outbreak," *IEEE Access*, vol. 8, pp. 163608–163617, 2020, doi: 10.1109/ACCESS.2020.3021983.
- [13] M. Goyal and R. Kumar, "The Pipeline process of signature-based and behavior-based malware detection," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, Oct. 2020, pp. 497–502. doi: 10.1109/ICCCA49541.2020.9250879.
- [14] V. S. P. Davuluru, B. Narayanan Narayanan, and E. J. Balster, "Convolutional neural networks as classification tools and feature extractors for distinguishing malware programs," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, Jul. 2019, pp. 273–278. doi: 10.1109/NAECON46414.2019.9058025.
- [15] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for Malware classification," in *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, Jul. 2016, pp. 338–342. doi: 10.1109/NAECON.2016.7856826.
- [16] J. Hegedus, Y. Miche, A. Ilin, and A. Lendasse, "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*, Dec. 2011, pp. 1016–1023. doi: 10.1109/CIS.2011.227.
- [17] D. O. Sahin, O. E. Kural, S. Akleyek, and E. Kilic, "New results on permission based static analysis for Android malware," in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, Mar. 2018, pp. 1–4. doi: 10.1109/ISDFS.2018.8355377.
- [18] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big data environment," *Applied Sciences*, vol. 12, no. 17, Aug. 2022, doi: 10.3390/app12178604.
- [19] A. A. Kardan, A. Kavian, and A. Esmaeili, "Simultaneous feature selection and feature weighting with K selection for KNN classification using BBO algorithm," in *The 5th Conference on Information and Knowledge Technology*, May 2013, pp. 349–354. doi: 10.1109/IKT.2013.6620092.
- [20] D. O. Sahin and S. Demirci, "Spam filtering with KNN: Investigation of the effect of k value on classification Performance," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, Oct. 2020, pp. 1–4. doi: 10.1109/SIU49456.2020.9302516.
- [21] T. Kumar, "Solution of linear and non linear regression problem by K nearest neighbour approach: By using three sigma rule," in *2015 IEEE International Conference on Computational Intelligence & Communication Technology*, Feb. 2015, pp. 197–201. doi: 10.1109/CICT.2015.110.
- [22] L. Chen, M. Li, W. Su, M. Wu, K. Hirota, and W. Pedrycz, "Adaptive feature selection-based AdaBoost-KNN With direct optimization for dynamic emotion recognition in human-robot interaction," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 2, pp. 205–213, Apr. 2021, doi: 10.1109/TETCI.2019.2909930.
- [23] R. Ghosh, S. Phadikar, N. Deb, N. Sinha, P. Das, and E. Ghaderpour, "Automatic eyeblink and muscular artifact detection and removal from EEG signals using K-nearest neighbor classifier and long short-term memory networks," *IEEE Sensors Journal*, vol. 23, no. 5, pp. 5422–5436, Mar. 2023, doi: 10.1109/JSEN.2023.3237383.
- [24] H. Al-Mimi, N. A. Hamad, M. M. Abualhaj, M. S. Daoud, A. Al-dahoud, and M. Rasmi, "An enhanced intrusion detection system for protecting HTTP services from attacks," *International Journal of Advances in Soft Computing & Its Applications*, vol. 15, no. 2, pp. 67–84, 2023.
- [25] M. A. Alsharaiah et al., "A new phishing-website detection framework using ensemble classification and clustering," *International Journal of Data and Network Science*, vol. 7, no. 2, pp. 857–864, 2023, doi: 10.5267/j.ijdns.2023.1.003.

## BIOGRAPHIES OF AUTHORS






**Prof. Mosleh M. Abu-Alhaj**    is a senior lecturer in Al-Ahliyya Amman University. He received his first degree in Computer Science from Philadelphia University, Jordan, in 2004, master degree in Computer Information System from the Arab Academy for Banking and Financial Sciences, Jordan in 2007, and Ph.D. in Multimedia Networks Protocols from Universiti Sains Malaysia in 2011. His research area of interest includes VoIP, Multimedia Networking, and Congestion Control. He can be contacted at email: m.abualhaj@ammanu.edu.jo.






**Dr. Ahmad Adel Abu-Shareha**    received his first degree in Computer Science from Al Al-Bayt University, Jordan, 2004, Master degree from Universiti Sains Malaysia (USM), Malaysia, 2006, and Ph. D degree from USM, Malaysia, 2012. His research focuses on Data mining, artificial intelligent and Multimedia Security. He investigated many machine learning algorithms and employed artificial intelligent in variety of fields, such as network, medical information process, knowledge construction and extraction. He can be contacted at email: a.abushareha@ammanu.edu.jo.






**Dr. Qusai Y. Shambour**    received the B.Sc. degree in Computer Science from Yarmouk University, Jordan, in 2001, the M.S. degree in computer networks from University of Western Sydney, Australia, in 2003, and the Ph.D. degree in software engineering from the University of Technology Sydney, Australia, in 2012. Currently, he is a Professor at the Department of Software Engineering, Al-Ahliyya Amman University, Jordan. His research interests include information filtering, recommender systems, VoIP, machine learning, and data science. He can be contacted at email: q.shambour@ammanu.edu.jo.



**Ms. Sumaya Nabil Alkhatib**    is a senior lecturer in Al-Ahliyya Amman University. She received his first degree in Computer Science from Baghdad University, Iraq, in June 1994 and master degree in Computer Information System from the Arab Academy for Banking and Financial Sciences, Jordan in February. Her research area of interest includes VoIP, Multimedia Networking, and Congestion Control. He can be contacted at email: sumayakh@ammanu.edu.jo.



**Mr. Mohammad O. Hiari**    is a lecturer in Al-Ahliyya Amman University. He received his first degree in Software Engineering from Philadelphia University, Jordan, in August 2004 and master degree in Computer Science from Al Balqa Applied University, Jordan in February 2016. His research area of interest includes VoIP, Multimedia Networking, and Congestion Control. He can be contacted at email: m.hyari@ammanu.edu.jo.