axioms

MDPI

Article

Applications of Fuzzy Logic and Probabilistic Neural Networks in E-Service for Malware Detection

Kristijan Kuk [1,*], Aleksandar Stanojević [2], Petar Čisar [1], Brankica Popović [1], Mihailo Jovanović [1], Zoran Stanković [3] and Olivera Pronić-Rančić [3]

[1] Department of Informatics and Computer Science, University of Criminal Investigation and Police Studies, 11080 Zemun, Serbia; petar.cisar@kpu.edu.rs (P.Č.); brankica.popovic@kpu.edu.rs (B.P.); mihailo.jovanovic@kpu.edu.rs (M.J.)
[2] Computer Emergency Response Team (CERT), Ministry of Interior of the Republic of Serbia, 11000 Belgrade, Serbia; aleksandar.zstanojevic@mup.gov.rs
[3] Faculty of Electronic Engineering, University of Niš, 18000 Niš, Serbia; zoran.stankovic@elfak.ni.ac.rs (Z.S.); olivera.pronic@elfak.ni.ac.rs (O.P.-R.)
* Correspondence: kristijan.kuk@kpu.edu.rs

Abstract: The key point in the process of agent-based management in e-service for malware detection (according to accuracy criteria) is a decision-making process. To determine the optimal e-service for malware detection, two concepts were investigated: Fuzzy Logic (FL) and Probabilistic Neural Networks (PNN). In this study, three evolutionary variants of fuzzy partitioning, including regular, hierarchical fuzzy partitioning, and k-means, were used to automatically process the design of the fuzzy partition. Also, this study demonstrates the application of a feature selection method to reduce the dimensionality of the data by removing irrelevant features to create fuzzy logic in a dataset. The behaviors of malware are analyzed by fuzzifying relevant features for pattern recognition. The Apriori algorithm was applied to the fuzzified features to find the fuzzy-based rules, and these rules were used for predicting the output of malware detection e-services. Probabilistic neural networks were also used to find the ideal agent-based model for numerous classification problems. The numerical results show that the agent-based management performances trained with the clustering method achieve an accuracy of 100% with the PNN-MCD model. This is followed by the FL model, which classifies on the basis of linguistic variables and achieves an average accuracy of 82%.

Keywords: cyber security; malware detection system; fuzzy logic (FL); agent-based technology; probabilistic neural network (PNN)

MSC: 68M25; 68T30; 62F86; 68T27; 68T07

check for updates

Citation: Kuk, K.; Stanojević, A.; Čisar, P.; Popović, B.; Jovanović, M.; Stanković, Z.; Pronić-Rančić, O. Applications of Fuzzy Logic and Probabilistic Neural Networks in E-Service for Malware Detection. *Axioms* **2024**, *13*, 624. https://doi.org/10.3390/axioms13090624

Academic Editors: Oscar Humberto Ross, Branislav Popovic, Marko Janev and Lazar Kopanja

Received: 14 July 2024
Revised: 8 September 2024
Accepted: 10 September 2024
Published: 12 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, there has been a rise in intelligent techniques-based research [1]. In the sphere of electronic commerce, the usage of agents in e-commerce applications is widespread [2]. Recently, there have been some notable advancements in the implementation of various intelligent strategies for constructing online intelligent e-services supported systems, where an intelligent online virus detection system is one of them. Not just in Serbia, as well as in the neighborhood, some organizations deal with internet-related incidents. That particular area of work is overseen by the Computer Emergency Response Team (CERT). They are a group of well-trained people who respond to computer incidents both in Serbia and internationally. When we consider the condition of informatics security in the region, we can see that not only our own country, but also our neighbors, face challenges. Serbia has committed to building a fundamental legal framework in this area with the approval of the Law on Information Security at the beginning of 2016 [3,4]. In Serbia, events can be reported through the "Academic Network of the Republic of Serbia" (AMRES) and SHARE CERT websites, but the Ministry of Interior's CERT does not have its website.

On the other side, online malware scanners are available to assist citizens who believe a virus has infected their machine or a file. One of the most widely used internet scanners is Virustotal [5]. It utilizes APIs (Application Programming Interfaces) from many free, open-source, and commercial antivirus engines to scan files for malware. Virus definition databases are updated regularly with official signatures released by the antivirus developers. While working, the website Virustotal checks the hash value of a file and its name to its antivirus definition databases. Unlike others, who rely on virus definition databases, our system does not yet employ databases but instead relies on the system's real-time scanning process and intelligent agent to detect threats.

Ch-J Lin et al. [6] employed a Convolutional Fuzzy Neural Network (CFNN) based on feature fusion and the Taguchi method to classify malware images, achieving satisfactory classification results. Hypergraph Neural Networks (HyperGNN) have also been frequently used for specific learning tasks, as these networks can encode high-order data correlations within a hypergraph structure. Zhang D. et al. [7] formalized Android malware detection as a hypergraph-level classification task. Their framework primarily involves modeling Android applications using hypergraph neural networks to represent various forms of topological relationships within these applications. The embedded features of Android applications are then used for malware detection. In the study by Atacak İ. [8], a Fuzzy Logic-Based Dynamic Ensemble (FL-BDE) model was proposed for detecting malware in the Android OS. The results indicate that the FL-BDE model outperformed traditional machine learning-based models significantly.

Further improvement in the intelligent malware detection e-service is obtained by using Probabilistic Neural Networks (PNN) that represent a special form of Radial Basis Function (RBF) networks and are used for numerous classification problems [9].

The paper is organized as follows: After the Introduction, Section 2 explains the implementation of a malware code detection e-service. In Section 3, an intelligent self-learning system based on fuzzy logic is proposed. The application of probabilistic neural networks in malware file detection is discussed in Section 4. Section 5 presents the most illustrative numerical results and discussion. Finally, the conclusions are provided in the last section.

2. Collection of Malware Behaviors on OS Windows

Many Government to Citizen (G2C) services collect and generate a wide variety of data and information, ranging from census data to scientific research. Many internet-based systems that use such online intelligent services have been developed in various domains of e-services to assist users and provide quality service in this regard. Various antivirus companies used to develop and sell online scanners to scan and detect malicious software and its symptoms in e-services. CERTs, in addition to them, play a significant role in the prevention of cyber-attacks. The Ministry of Interior's CERT handles malicious code reports and coordinates the country's other computer incident response teams. Its mission is to promote awareness, advise, and safeguard the Republic of Serbia's ICT system from cyber security threats. The main goal is to become more resilient to and prevent various types of cyberattacks, including malware, viruses, worms, denial of service (DOS), etc. [10], so that citizens can easily recognize them when the Ministry of the Interior offers such a service on the e-services.

Our intelligent system's main purpose is to gather data from potentially risky system places, such as processes and memory, that may contain covert malicious code. Users can download the online application from the web page thanks to the developed e-service. The user scans his machine after starting the application. He will be prompted to send his scanned results to CERT after the scan is completed. He then completes the form and sends the scanned findings from his computer to the e-service. After receiving the results, the agent compares them to the threat level in the learned database before deciding whether to notify the CERT staff or not.

The web server programming language PHP is used to identify fresh impending reports for the technological realization of the intelligent malware detection e-service in the suggested service. It detects the server's incoming reports, downloads the parameters from the attached file (which contains the scanned data), converts the numeric values into linguistic variables, and then determines the rules for the intelligent agents' actions based on the levels of danger, which are then visually displayed using standard CSS techniques. Without using prepackaged modules that can be found in several agent platforms for certain programming languages, a whole agent of logic and technique can be built using regular PHP, HTML, and CSS scripts. We gain a state of real dangers to the computer system using agent-based technologies. We need genuine continuous values to determine the amount of true threat to identify the degree of danger.

Detecting malware using Process Explorer can be a tedious process, but this tool enables users to search for malicious processes within the Windows operating system quickly. Monitoring various memory metrics during runtime is crucial for understanding significant memory usage. Process Explorer provides valuable metrics such as WS Private, WS Shareable, and WS Shared, which are instrumental in malware detection and analysis [11]. Additionally, Microsoft Defender Antivirus can be used to block potentially harmful applications on the system. Through the correlation between analyzed malware and non-malware processes using Process Explorer and Microsoft Defender Antivirus, noticeable changes were observed in the following parameters: Data Execution Prevention, Integrity, Paged Pool, WS Shareable, and WS Shared.

If a program attempts to execute code from memory incorrectly, the Data Execution Prevention (DEP) parameters will terminate it. DEP consists of three levels of states: disabled, disabled (permanent), and enabled (permanent). Its primary objective is to safeguard our system from malicious code and other security threats. The Integrity values represent a security mechanism introduced in Windows Vista and can exist in five states: Untrusted, Low, Medium, High, and System. This classification is important because a higher integrity level grants a process greater capabilities. For instance, processes with low-level integrity operate within app containers, while applications launched by standard users typically have a medium integrity level. Administrators are assigned a high integrity level, whereas kernel processes and services operate at the system level. Furthermore, when considering the values of I/O writes, we can determine the number of input/output operations performed. A significant number of writing operations may indicate the presence of malicious code. If an unknown application exhibits a continuous pattern of high write operations, it raises our suspicion of high-risk malicious activity. Our analysis indicates that the range of I/O write operations typically varies from 0 to 1800.

Discovering knowledge about the process in the database to identify and delete as much irrelevant information as possible is the process of selecting a subset of attributes [12]. This is particularly important in our case as future fuzzy-based rules may lead to 3^n combinations. For example, with 8 input variables, there may be a total of 6561 possible rules, resulting in a significant number of combinations for an agent to process. We analyze the data by calculating Pearson's correlation between input and output [13]. In intelligent technologies, measures of information gain are often used to indicate the degree of mixing between features, as shown in [14]. Based on the Pearson correlation of the malicious features analyzed, we obtained the following results (Table 1):

Table 1. Malicious features on OS Windows via information gain.

Information Gain	Features
0.82029	Integrity
0.73754	WS Shared
0.71231	I/O Writes
0.6879	Disk Writes
0.6451	DEP

The results show that certain malicious features have a significant impact on malware detection. To eliminate redundancy, we selected a subset of features for which all pairwise Pearson correlations between the input variables in the subset were above 0.50. The highest information gain was used as the criterion for selecting informative features. Therefore, only 5 features were selected: Integrity, WS Shared, I/O Writes, Disk Writes, and DEP.

We concluded that the values derived from this process should be considered malicious characteristics of the Windows operating system [15]:

- Integrity—represents a security mechanism that protects the components of the Windows security architecture and restricts access permissions of applications that run from the same user account.
- Disk Writes—displays how many different write sizes there are on the local storage.
- I/O Writes—the total number of input/output operations of the write process, including file, network, and device I/Os.
- WS Shared—shows the amount of shared memory a process uses in the set of pages in the virtual address space which is currently occupied in the physical memory.
- Data Execution Prevention—Windows security feature that monitors the safe and secure use of system memory. Its primary objective is to safeguard the system from malicious software and other security risks.

3. Developing the Fuzzy-Based Model

Agent-based technologies are generally used in our everyday communication. Discrete values should be used to build intelligent agents. We plan to turn discrete values into conditions of linguistic phrases based on fuzzy logic by feeding information to a discretized with outputs from the intelligent malware detection e-service (a standard fuzzy space of three sets: LOW, MEDIUM, and HIGH). The linguistic phrase is measured using an appropriate function [16,17]. The form of the membership function is important in fuzzy-based systems because it affects the system overall. Different shapes are possible, like triangular, trapezoidal, Gaussian, etc. The triangular shape was used in our situation. The triangular membership function is determined by three parameters {a, b, c} using minimum and maximum in the following way:

$$triangle(x; a, b, c) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right). \quad (1)$$

We examined a computer system infected with malicious code to obtain the maximum and minimum input variables.

3.1. Fuzzy Partition Design

Assuming that linguistic variables are well-known, we define expert knowledge. To subsequently facilitate the process given by a possible small number of linguistic variables, the specialists identify the properties' determinations, such as the domain of interest within the physical range. As stated above, we assume that the minimum information about the membership function definition is available, including many terms, sometimes prototypes of linguistic labels (modal points), and the definition of universes. Automatic database design is one of the most important steps in defining the entire knowledge base. This can be concluded from most of the existing approaches [18], which focus on the generation of fuzzy partitions from data, as discussed in [19].

By defining the optimal number of linguistic terms in a fuzzy partition and within the universe of the variable where the fuzzy set is located, the best form of the function is determined. This process is based on the automatic generation of fuzzy partitions. The evolution of fuzzy partitions is based on partitioning and linguistic properties. The relationship between fuzzy sets corresponding to the same variables influences the form of fuzzy sets, which are presented as different dissimilarity measures [20]. Several indices have been defined to characterize fuzzy partitions. The degree of membership of the k-th

element from the data set to the *i*-th element of the fuzzy partition is indicated by the distance measure at the partition level [21]. The partition coefficient (*PC*) is defined as:

$$PC = \frac{\sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^2}{n} \tag{2}$$

where u_{ik} is the membership value for the *k*-th pattern in the *i*-th cluster.

The partition entropy (*PE*) is defined as:

$$PE = -\frac{1}{n}\left\{ \sum_{k=1}^{n} \sum_{i=1}^{c} [u_{ik} \log(u_{ik})] \right\} \tag{3}$$

where *c* is the number of elements of the fuzzy partition and *n* represents the cardinality of the set of data. A good partition should minimize entropy and maximize the partition coefficient.

Additionally, the following index measure called the Chen index (*CI*), was introduced [22]:

$$CI = \frac{1}{n} \sum_{k=1}^{n} \max(u_{ik}) - \frac{2}{c(c-1)} \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} \frac{1}{n} \sum_{k=1}^{n} min\left(u_{ik}, u_{jk}\right). \tag{4}$$

For a partition to be efficient, it should minimize the entropy while maximizing the partition coefficient and the Chen index. The reasoning mechanism that fosters decision-making rules is the most important part of the knowledge-based system. For our fuzzy-based model, we proposed two approaches for the fuzzy partition initialization. First, the purely automatic induction method (K-means, Hierarchical Fuzzy Partitioning, and Regular Partitioning) and the approach derived from the extraction of information from textual sources. Evaluation of unclear partitions implies two different aspects: language properties and partitioning characteristics. Language properties are exclusively related to the form of fuzzy sets and the relations between the fuzzy sets defined for the same variable. Their assessment does not include data. On the other hand, partition properties include the level of alignment of the partition with the data from which the partitions are derived. It applies only to partitions derived from data. Two fundamental issues to be addressed in a typical cluster scenario are:

1. "How much cluster is actually present in the data?"
2. "How real are linguistic grouping (labels) for the fuzzy set?"

For the first question, we assume that the number of clusters present in the data is equal to the number of classes. In this section, we have studied external criteria for answering the second question. To examine the performance of external indexes introduced in the previous sections, we have applied the most common fuzzy algorithms of the grouping: "HFP", "Regular" and "K-means". These algorithms have been applied to 4 malicious features: Disk Writes (DWR), Integrity (INT), I/O Writes (IOW) and WS Shared (WSS). Data Execution Prevention (DEP) parameter in Windows contains 4 levels of states: Disable, Disable (permanent), Enable and Enable (permanent) and we have not applied any fuzzy algorithm to that feature. We did not include these two parameters in the fuzzy algorithm for that feature.

Fuzzy Membership Function

A fuzzy set *A* is a pair (U, μ_A) where *U* is a reference set and μ_A: $U \rightarrow [0, 1]$ is a membership function [23]. The reference set *U* is called the universe of discourse, and for each *x* belonging to U, the value $\mu_A(x)$ is called the grade of membership of *x* in $(U, \mu_A\}$.

Definition 1. *If X is a collection of objects denoted generically by x, then a fuzzy set A in U is a set of ordered pairs [24]*:

$$A = \{(x, \mu_A(x))|x \in U\}. \quad (5)$$

The translation of fuzzy quantity into crisp quantities is called defuzzification d(A). The defuzzified value of A by the signed distance is given by:

$$\mathrm{d(A)} = \frac{1}{2}\int_0^1 [(A_L(\alpha) + A_R(\alpha))]\mathrm{d}\alpha \quad (6)$$

where $A_L(\alpha)$ is the left distribution of the confidence interval, $A_R(\alpha)$ is the right distribution of the confidence interval of the defuzzification number A and α ∈ [0,1].

If $A = (a_1, a_2, a_3)$ is a triangular fuzzy number, then $A_L(\alpha) = a_1 + (a_2 - a_3)\alpha$, and $A_R(\alpha) = a_3 - (a_3 - a_2)$. The signed distance of a triangular fuzzy number $d(A) = (a_1, a_2, a_3)$ can be calculated as:

$$\mathrm{d(A)} = \frac{1}{4}(a_1 + 2a_2 + a_3). \quad (7)$$

If $A = (a_1, a_2, a_3, a_4)$ is a trapezoidal fuzzy number, then $A_L(\alpha) = a_1 + (a_2 - a_3)\alpha$, and $A_R(\alpha) = a_4 - (a_4 - a_3)$. The signed distance of a trapezoidal fuzzy number $d(A) = (a_1, a_2, a_3, a_4)$ can be calculated as:

$$\mathrm{d(A)} = \frac{1}{4}(a_1 + a_2 + a_3 + a_4). \quad (8)$$

Fuzzy sets can have a variety of shapes. However, a triangle or a trapezoid number provides an adequate representation of the expert knowledge and at the same time, significantly simplifies the process of computation. Triangular fuzzy numbers (TFN) and trapezoidal fuzzy numbers (TrFN) are used very often to deal with the vagueness of the decisions related to choice for each of the criteria [25]. It is easy to observe that the TFN (a_1, a_2, a_3) can be considered as a special case of the TrFN (a_1, a_2, a_3, a_4) with $a_3 = a_2$.

One of the steps in the process of developing a fuzzy system is specifying the problem and defining linguistic variables. If a variable can take words in natural languages as its values, it is called a linguistic variable, where the words are characterized by fuzzy sets [26]. In the process of fuzzification, the main problem is how to determine the number of linguistic variables. It depends on the level of distinction. In practice, 3, 5, or 7 linguistic variables are used which are needed to describe some problem. If we use more variables, then we have higher accuracy. Sometimes, we have to think about what accuracy is needed. The best approach is to consult an expert; however, that person is not always available. It is necessary to automate the best algorithm based on rules to select an index and track its evolution throughout the optimization process. Measuring the clustering performance of different algorithms is crucial, as larger PC and smaller PE values indicate a better partition. If this is not the case, the CI index can be helpful. Table 2 shows the fuzzy partition quality for each malicious feature. From this table, we can draw the following conclusions:

DWR feature:

- Using the HFP method, the best results were achieved using 2 variables (labels) during the validation process, with a PC of 0.97352, PE of 0.04460, and CI of 0.96576
- Using the Regular method, the best results were obtained with 2 variables (labels) during the validation process, yielding a PC of 0.99282, PE of 0.01539, and CI of 0.99185.
- Using the K-means method, the best results were also achieved with 2 variables (labels) during the validation process, showing a PC of 0.99417, PE of 0.01141, and CI of 0.99325.

From this, we can conclude that the optimal result would be the PC achieved with the K-means algorithm using 2 variables (labels).

IOW feature:

- Using the HFP method, the best results were achieved using 2 variables (labels) during the validation process, with a PC of 0.99538, PE of 0.00848, and CI of 0.99399.
- Using the Regular method, the best results were obtained using 2 variables (labels) during the validation process, with a PC of 0.99578 and PE of 0.00785. However, the CI yielded the best result (0.99664) using 3 variables.
- Using the K-means method, the best result was obtained using 3 variables (labels) during the validation process with a PC of 0.99662. The best result with 2 variables was achieved using PE (0.00648). For the CI, the best result (0.99781) was achieved using 3 variables.

From this, we can conclude that the optimal result would be the CI achieved with the K-means algorithm using 4 variables (labels).

WSS feature:

- Using the HFP method, the best results were achieved using 2 variables (labels) during the validation process, with a PC of 0.85821 and PE of 0.20514. However, the best CI value (0.86953) was obtained using 5 variables.
- Using the Regular method, the best results were obtained using 5 variables (labels) during the validation process, yielding a PC of 0.84533, PE of 0.23687, and CI of 0.88384.
- Using the K-means method, the best results were also achieved using 5 variables (labels) during the validation process, with a PC of 0.86365, PE of 0.20999, and CI of 0.89743.

From this, we can conclude that the optimal result would be the CI achieved with the K-means algorithm using 5 variables (labels).

Table 2. Fuzzy partition quality (considering the number of linguistic variables n_{LV}).

Features	Algorithms	PC	n_{LV}	PE	n_{LV}	CI	n_{LV}
DWR	HFP	0.97352	2	0.04460	2	0.96576	2
	Regular	0.99282	2	0.01539	2	0.99185	2
	K-means	0.99417	2	0.01141	2	0.99325	2
IOW	HFP	0.99538	2	0.00848	2	0.99399	2
	Regular	0.99578	2	0.00785	2	0.99664	3
	K-means	0.99662	3	0.00648	2	0.99781	4
WSS	HFP	0.85821	2	0.20514	2	0.86953	5
	Regular	0.84533	5	0.23687	5	0.88384	5
	K-means	0.86365	5	0.20999	5	0.89743	5

From the above table, we can now set the number of labels for each set of malicious feature values. By the K-means algorithm with max CI, the features give the following linguistic variable values: DWR—2 variables, IOW—4 variables, and WSS—5 variables.

Fuzzy logic is used for the description of the possible values of variables that most affect the detection of malware (DWR)—for the determination of low, and high values. Namely, the curves for each variable are created and their generalization (creating a common general form) is made in the following two steps:

- The characteristic points A, B, C, and D are detected on the x-axis since there are 5 observed variables. 5 corresponding values are read for each point (e.g., A1–A5, B1–B5, etc.)
- For each point the mean value is calculated as:

$$X_{avg} = \sum_{i=1}^{5} \frac{X_i}{5} . \tag{9}$$

In that way, the generalized fuzzy curve has a trapezoidal shape and is determined by the points A_{avg} (Very Low) = 0.15925, B_{avg} (Low) = 0.40625, C_{avg} (Medium) = 0.5335 and D_{avg} (High) = 0.84. Figure 1 presents a general form for linguistic variables with a trapezoidal shape for malicious features.

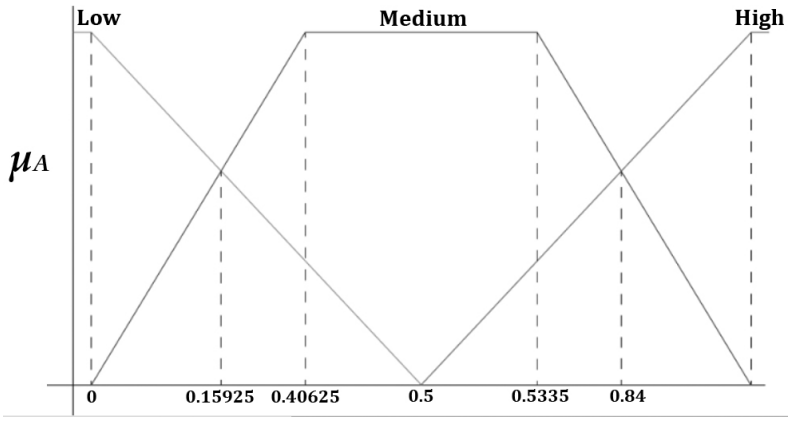

Figure 1. General form fuzzy set in a trapezoidal shape.

Following the methodology developed in [26], in this work, we propose a modified algorithm for the extraction of μ_A using TFN and TrFN numbers. The Algorithm 1 was implemented through the following steps:

Algorithm 1 For the extraction of function μ_A

Set of n points x_i: $n_{\text{points}} = \{x_1, x_2, \ldots, x_n\}$;
A point scale of a finite number of Linguistic Terms (LT_s):
$LT_s = \{LT_1, LT_2, \ldots, LT_m\}$;
Each LT has an associated TrFN $T_l = (a_{l1}, a_{l2}, a_{l3}, a_{l4})$:
TrFNs = $\{T_1 = (a_{11}, a_{12}, a_{13}, a_{14}), \ldots, T_m = (a_{m1}, a_{m2}, a_{m3}, a_{m4})\}$;
For each x_i in n_{points} collect n TrFNs:
$T_{ik} = (a_{1ik}, a_{2ik}, a_{3ik}, a_{4ik})$, $k = 1, 2, \ldots, n$;
Assemble TrFNs for each x_i:
For $i = 1$ to n:
1. $a_{1i} = min\{a_{1ik}\}$
2. $a_{2i} = a_{1i} + \frac{1}{n}\sum_{k=1}^{n} a_{2ik}$
3. $a_{3i} = a_{4i} - \frac{1}{n}\sum_{k=1}^{n} a_{3ik}$
4. $a_{4i} = max\{a_{4ik}\}$
5. $T_i = (a_{1i}, a_{2i}, a_{3i}, a_{4i})$

For each T_i in $\{T_1, T_2, \ldots, T_n\}$ convert T_i to risk y_i, whose values are $0 \leq y \leq 1$:
$y_i = T_i / \sum_{k=1}^{N} T_{ik}$.

Using Lagrange's interpolation polynomial formula, we can see that the values on the y-axis, represented by a trapezoidal shape, are not uniformly distributed. Based on Lagrange polynomials, we arrive at a simple quadratic representation of the distance data. This algorithm aims to find the membership polynomial using Lagrange's polynomial formula, where μ_A represents the relative distance compared to the lowest average distance (A_{avg}, B_{avg}, C_{avg}, D_{avg}) obtained from the given fuzzy sets. For five linguistic variables, it is optimal to use a trapezoidal fuzzy number for the value 0.628 of the fuzzy membership function. Based on this value of the fuzzy membership function (μ_A), the quality of the fuzzy partition (linguistic variable), and the shape of the fuzzy curve, the malicious features are assigned threshold values for the possible output linguistic labels for DWR, IOW, and WSS. On the other hand, the parameters DEP and INT have predefined output

values from the operating system. Table 3 provides an overview of all possible outputs for all parameters.

Table 3. The output linguistics labels for all malicious features.

DEP	*INT*	Threshold Values	*DWR*	*IOW*	*WSS*
Disable	Untrusted	[0–0.15925]	Low	Low	Very Low
Disable (permanent)	Low	[0.15926–0.40625]	Low	Medium	Low
Enable	Medium	[0.40626–0.5335]	Low	Medium	Medium
Enable (permanent)	High	[0.5336–0.84]	High	Medium	High
	System	[0.85–1]	High	High	Very High

3.2. Creating Fuzzy Rules Based on the Apriori Algorithm

The development of intelligent agents, mobile agents, and other Apriori technologies has recently gained attention [27], as these technologies simplify the interaction between users and computers. Some related works have explored intelligent agent solutions in e-government using multi-agent systems [28]. In contrast, we employ the most basic type of agent in our architecture. After the user has completed the form and sent the computer scan findings to the e-service, and we've obtained the final danger parameters, the agent's goal in the recommended system is to select whether to publish or deny the threat to the Ministry of Interior.

The intelligent malware detection e-service generates a variable called "danger" with two possible values: "yes" or "no", which is defined by an agent set. When malware of the domain was detected on the machine as a virus, we increased the number of instances in the current database accordingly. Weka [29] used the total amount of acquired data (200) to create agent rules using the Apriori algorithm. The goal of this approach was to identify the rules of the form $X \rightarrow Y$, where X and Y are items or item sets. This means that a database instance that contains X is also likely to contain Y. To evaluate the performance and efficiency of the algorithm, we consider two key parameters: Support (*Supp*) and Confidence (*Conf*), [30,31]. The support for the rule $X \rightarrow Y$ indicates the probability or frequency with which the item sets X and Y occur together in instances of the database. The confidence indicates the probability that Y occurs in a database instance if X is already present. A high confidence indicates that the rule is reliable. These parameters are therefore defined as follows:

$$\begin{array}{c} Supp(X \rightarrow Y) = P(X \cap Y) \text{ and} \\ Conf(X \rightarrow Y) = P(Y|X) = \frac{P(X \cap Y)}{P(X)} \end{array} \quad (10)$$

where P is the probability that an item is present in a set of frequent items.

In our scenario for the proposed intelligent malware detection e-service that can detect hidden malicious behaviors (e.g., cyber espionage and camera surveillance) in Windows operating systems, we set the minimum support for the created rules to 0.45. This means that items or groups of items must occur in at least 45% of the observed instances (i.e., 90 out of 200 instances). In contrast, we defined the minimum confidence as 0.1, indicating a probability of more than 10% that items or groups of items occur together in the instances.

Examining the association rules in Equation (10), we observe several fuzzy-based rules. These fuzzy-based rules for malware detection are presented in Table 4. For example, if the Integrity level is Medium, the process is not considered suspicious. Conversely, if *INT* is High and *WSS* is Very Low, *IOW* is Low, or *DW* is Low, these conditions impact the system and trigger an alert, indicating that something may be wrong and requires caution. Consequently, we have an intelligent agent rule based on a basic IF-THEN rule, which includes a condition and a conclusion.

Table 4. Fuzzy-based rules for malware detection.

No.	IF-THEN Rule
1.	IF (*INT* is Medium) THEN command is No
2.	IF (*INT* is High) AND (*WSS* is Very Low) THEN command is Yes
3.	IF (*INT* is High) AND (*IOW* is Low) THEN command is Yes
4.	IF (*INT* is High) AND (*DW* is Low) THEN command is Yes

Our proposed intelligent agent-based management system, built on the aforementioned rules, is designed to assist CERT in identifying potential attack types through an e-form, indicating a possible threat to a citizen's computer. A special web page managed by CERT contains all the relevant information about the citizen who submits the report, which is then reviewed by a police officer at CERT. By utilizing fuzzy logic for agent-based rules, the system supports decision-making by detecting malicious code and determining whether to alert CERT staff (command Yes) or not (command No).

4. Probabilistic Neural Network Model

Further improvement in malware code detection (MCD) is obtained by using Probabilistic Neural Networks (PNN). PNNs have been widely used in different scientific areas for pattern recognition problems [32–36].

The primary task of the developed PNN-MCD model is to decide whether the analyzed process running in the system represents a malware or non-malware process. The decision is based on the values of the following parameters that describe the current behavior of the process: DEP (*DEP*), Integrity (*INT*), WS Shared (*WSS*), I/O Writes (*IOW*), and Disk Writes (*DWR*). This decision-making process can be viewed as a classification problem, where the first class consists of processes that do not represent malware, while the second class consists of processes that do represent malware. The architecture of the proposed model is shown in Figure 2. The proposed PNN-MCD model consists of a PNN-MCD network and a CtoM module.

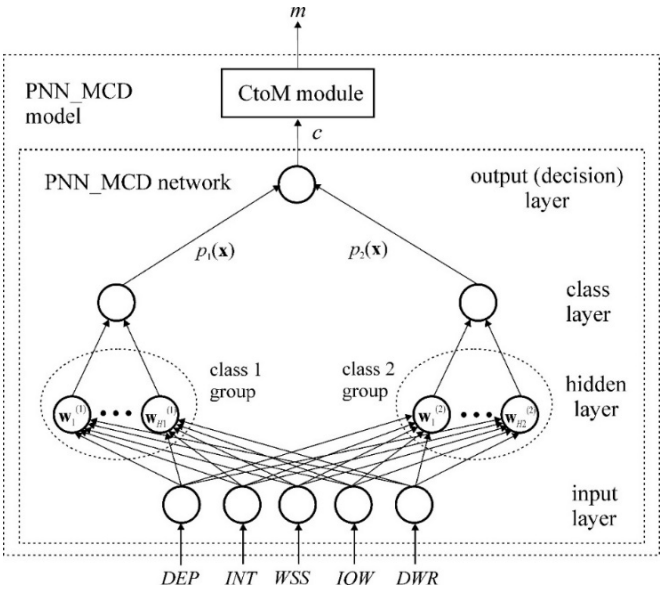

Figure 2. Architecture of PNN-MCD model.

Let a sample of the process represent an ordered combination of the current values of the *DEP*, *INT*, *WSS*, *IOW*, and *DWR* parameters, which describe the observed process at the moment these parameters are sampled. Let the sample be represented by the input variable vector $\mathbf{x} = [DEP\ INT\ WSS\ IOW\ DWR]^T$, which is fed into the input of the PNN-MCD network. Since the *DEP* and *INT* parameters in their original form are assigned symbolic

values (Table 3) with which the PNN-MCD network cannot work, it is necessary to convert these values into real values that can be fed to the input of the neural network. This was done by assigning the symbols a real value from the range [0,1] where every two adjacent symbols on the scale from Table 3 are at the same distance in the [0,1] space. Thus, the mapping of the symbolic values of the DEP parameter into real values is given in the form {('Disabled permanent', 0), ('Disabled', 0.33), ('Enabled', 0.66), ('Enabled permanent', 1)}, while is the mapping of the symbolic values of the *INT* parameter into real values given in the form {{('App container', 0), ('Low', 0.25), ('Medium', 0.5), ('High', 0.75), ('System', 1)}.

The PNN-MCD network is tasked with classifying a sample $\mathbf{x}$ into either class 1 ($c = 1$, where c represents the class index)—corresponding to non-malware process samples, or class 2 ($c = 2$)—corresponding to malware process samples. In other words, this classification problem can be represented as mapping the sampled continuous parameter values ($\mathbf{x}$) into discrete values of the class index (c) as follows:

$$c = f_{\text{PNN-MCD}}(\mathbf{x}) = f_{\text{PNN-MCD}}([DEP\ INT\ WSS\ IOW\ DWR]),\ \ c \in \{1,2\}. \tag{11}$$

The input layer of the PNN-MCD network (buffer layer) forwards the values of the input vector $\mathbf{x}$ to each neuron in the hidden layer. The number of neurons in the input layer, M, is equal to the number of input variables, i.e., equal to the size of the vector $\mathbf{x}$, $M = 5$.

The hidden layer of the PNN-MCD network is the information carrier about the classes. Neurons in the hidden layer are divided into two groups so that each class c ($c = 1, 2$) has its group of neurons. The number of neurons within each group H_c, $c = 1, 2$, is equal to the number of samples in the training set that belongs to class c. H_c is determined during the training phase of the neural network. The activation function of neurons in the hidden layer that belong to class c is based on the Gaussian function [9] and the outputs of neurons of class c are given as:

$$h_i^{(c)}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{M}{2}}\sigma^M} \cdot e^{-\frac{\|\mathbf{x}-\boldsymbol{w}_i^{(c)}\|}{2\sigma^2}},\ i = 1, 2, \ldots, H_c \tag{12}$$

where vector $\boldsymbol{w}_i^{(c)}$ of dimension M represents the vector of the center of activation function of i-th neuron that belongs to class c, and σ is the spread parameter (standard deviation) of the activation function [9,32].

The neurons in the class layer serve to sum the outputs of the neurons in the hidden layer separately for each class and, based on this summation, to estimate the probability that sample $\mathbf{x}$ belongs to the considered class. Each group of neurons of class c, $c = 1, 2$ from the hidden layer corresponds to one neuron in the class layer, therefore the total number of neurons in the class layer is 2. The estimation of the probability that sample $\mathbf{x}$ belongs to class c, $p_c(\mathbf{x})$, is performed by the c-th neuron in the class layer through its activation function based on the Parzen window technique [34,36], so the outputs of this layer are given as:

$$p_c(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{M}{2}}\sigma^M} \cdot \frac{1}{H_c}\sum_{i=1}^{H_c} e^{-\frac{\|\mathbf{x}-\boldsymbol{w}_i^{(c)}\|}{2\sigma^2}},\quad c = 1, 2\,. \tag{13}$$

The output layer (also called the decision layer) contains a single neuron responsible for determining the class to which the input sample most likely belongs, based on the estimated probabilities provided by the class layer. The output neuron makes this decision by evaluating the outputs of all the neurons in the class layer, following Bayes's decision rule. The class with the highest estimated probability is selected as:

$$c = \arg\max_c\{p_c(\mathbf{x})\},\quad c = 1, 2 \tag{14}$$

where $\arg\max_c$ denotes the class c that corresponds to the highest probability $p_c(\mathbf{x})$.

For training the PNN-MCD network, a training set of samples is used, where each sample (a combination of input variable values) is assigned a desired (target) class in-

dex value that is expected at the output of the trained network when the corresponding sample is provided at the network's input. Accordingly, the training set P can be represented as follows $P = \left\{ (\mathbf{x}_1^P, c_1^{DP}), (\mathbf{x}_2^P, c_2^{DP}), \ldots, (\mathbf{x}_s^P, c_s^{DP}), \ldots, \left(\mathbf{x}_{N_P}^P, c_{N_P}^{DP}\right) \right\}$ where c_s^{DP} is the desired value at the network output for the sample x_s^P at the network input and N_P represents the total number of training samples. To test the trained network, a test set of the same format as the training set is used, that does not contain any samples used in the training process. Therefore, the training set can be represented as follows $T = \left\{ (\mathbf{x}_1^T, c_1^{DT}), (\mathbf{x}_2^T, c_2^{DT}), \ldots, (\mathbf{x}_s^T, c_s^{DT}), \ldots, \left(\mathbf{x}_{N_T}^T, c_{N_T}^{DT}\right) \right\}$ where c_s^{DT} is the desired value at the network output for the sample x_s^T at the network input and N_T represents the total number of test samples.

The numbers of neurons in the hidden and class layers of the PNN-MCD network are determined during the network training. Also, the weighting vectors in the hidden layer are adjusted so that the neural network performs the correct classification of all samples from the training set. The spread parameter is not determined during the network training as its value is set before the training. The spread parameter value has an impact on the generalization capabilities of the PNN-MCD network. That impact can be quantified through the percentage of correctly classified samples during the testing phase on the set of samples not used for the network training. By multiple repetitions of network training for different values of the spread parameter and by evaluating the network performances during the testing phase, the value of the spread parameter can be adjusted so that the number of incorrectly classified samples during the testing of the network is as small as possible.

Using a PNN instead of traditional deep neural networks with a softmax layer as the final layer offers the advantage of significantly accelerating the development of neural models for malware detection. This is because it eliminates the need for complex and slow training algorithms based on iterative gradient descent, which involves managing numerous hyperparameters (such as learning rate, momentum, and batch normalization parameters). In contrast, training a PNN is simpler and performed in parallel with network construction, focusing on a single hyperparameter: the spread parameter (σ). During the construction and training of the PNN, each sample $\mathbf{x}_s$ from the training set P is assigned to a hidden neuron in the group of neurons for class ${c_s}^{DP}$, with the weight of that neuron set to the value of $\mathbf{x}_s$.

The CtoM module has the task of mapping the class index value c to the logical statement of the process status, i.e., to the logical variable m which shows whether the observed process is malware (m = Yes) or non-malware (m = No). The mapping function of the CtoM module ($c \rightarrow m$) is defined in Table 5.

Table 5. Mapping function of CtoM module ($c \rightarrow m$).

Class index (c)	1	2
Process status (m)	No	Yes

5. Experimental Results

A validity (accuracy) check was performed on the basic form rules based on fuzzy logic alignment, comparing them to the actual prediction. We aim for the classifier to identify positive and negative samples correctly. To evaluate the performance of both models in agent-based management, we use the accuracy rate (AR), defined as:

$$AR = \frac{(TP + TN)}{(TP + FP + FN + TN)}, \tag{15}$$

where TP (True Positive)—are instances where malware is appropriately identified by the agent as malicious code (m = YES, true test-positive); FP (False Positive)—the scenarios where there is no malware but the agent incorrectly detects it as malicious code (m = YES,

false test-positive); *FN* (False Negative)—scenarios where the agent incorrectly recognizes a malware while there is no dangerous code (*m* = NO, false test-negative); *TN* (True Negative)—the scenarios in which the agent properly recognizes the absence of harmful code (*m* = NO, true test-negative).

A total of 50 samples were analyzed in the FL model, with 25 samples containing harmful code and 25 containing no malicious code. These rules were derived from comparing the infected malware with non-infected malware processes on a virtual machine. The following results are obtained: *TP* = 25, *FP* = 9, *TN* = 16, *FN* = 0 (*Precision* = 0.868, *Recall* = 0.820, *F-Measure* = 0.814, *MCC* = 0.686, and *ROCArea* = 0.740), indicating that our system is 82% accurate (*AR* = 0.82), and we believe that the agent-based management with the FL model has good precision in cases of cyber-attacks in an intelligent malware detection e-service with that level of precision.

With the aim of further accuracy improvement in malware code detection, we developed the PNN-MCD model. For the development of the PNN-MCD neural network, we used a set containing 200 samples. 150 of them are used for PNN-MCD neural network training, while the rest of the 50 samples (the same instances for testing in the FL model) are used for network testing. The lower and upper bounds of the range within which the values of the input parameters vary for the samples from the training and test sets are given in Table 6.

Table 6. Ranges of input parameters for samples from the training and test set.

Parameter	Min	Max
DEP	0	1
INT	0	1
WSS	0	102,076,416
IOW	0	1,806,728
DWR	0	10,614

To estimate the accuracy of the presented PNN-MCD model, we investigate the dependence of the PNN-MCD model accuracy rate (*AR*) on the spread parameter (σ). The spread parameter is changed in a wide range $\sigma \in [10^{-3}–10^{10}]$. The results obtained for the train and test set are shown in Figure 3. The PNN-MCD neural network training with the spread parameter values in the range $8619 \leq \sigma \leq 244{,}109$ provides a maximum value of *AR* (*AR* = 1, *TP* = 25, *FP* = 0, *TN* = 25, *FN* = 0) for both the training set (shown by dashed line) and the test set (solid line). In other words, the specified range of the spread parameter defines the operating range of the PNN-MCD model since the neural network training using the spread parameter from that range gives a model that shows the ideal classification for both the train and test set.

The training of the PNN-MCD neural network is performed for different values of spread parameters for each training session. After the end of each training session, the quality of training and the generalization capabilities of the trained network were examined by its testing on the set of samples not used for the training and by calculating the accuracy of the model in the process of classifying samples according to the Equation (15).

To confirm that the testing results obtained for the fuzzy logic and PNN-MCD models on the test set T are valid and not due to random variation in the input data, a statistical performance analysis of both models was conducted. The analysis included the following steps:

- An analysis of classification accuracy (*AR* values) on two randomly selected, mutually disjoint subsets of the test set T, to examine the consistency of both models on different data subsets.
- A *p*-value analysis, using a paired *t*-test (separately for the fuzzy logic model and the PNN-MCD model) on the two different aforementioned subsets, to assess the stability and robustness of both models on different samples from the same data distribution.

- A *p*-value analysis, using an independent *t*-test for the fuzzy logic model and PNN-MCD model on the same randomly selected subset from set T, to determine whether there is a statistically significant difference in the performance of the two models and, if so, whether this difference is potentially the result of random variation in the input data.

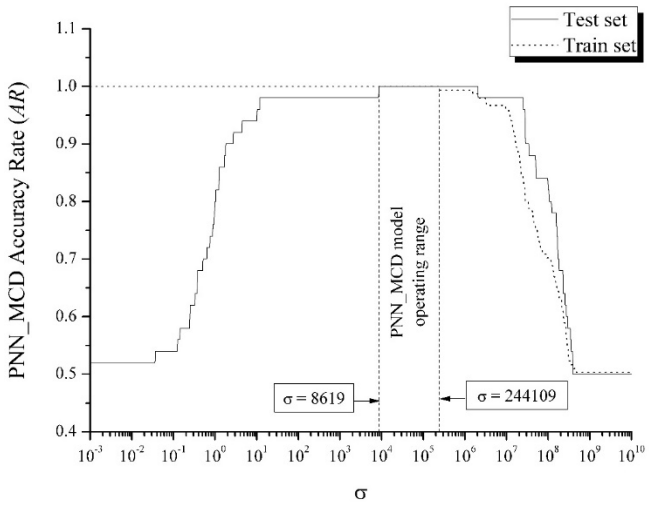

Figure 3. PNN-MCD model accuracy rate versus spread parameter.

Let the test set T be randomly divided into two mutually disjoint sets, R and Q, with N_R and N_Q elements, respectively, where $N_T = N_R + N_Q$ (the number of elements of set R, N_R, is randomly selected from the range [1, card(T)-1], while the number of elements of set Q, N_Q, is calculated as card(T)-N_R).

Let $\mathbf{e}_{FL}{}^R = [e_{FL}(\mathbf{x}_1{}^R), e_{FL}(\mathbf{x}_2{}^R), \ldots, e_{FL}(\mathbf{x}_{NQ}{}^R)]$ represent the error vector of the fuzzy logic model on set R, and $\mathbf{e}_{PNN\text{-}MCD}{}^R = [e_{PNN\text{-}MCD}(\mathbf{x}_1{}^R), e_{PNN\text{-}MCD}(\mathbf{x}_2{}^R), \ldots, e_{PNN\text{-}MCD}(\mathbf{x}_{NQ}{}^R)]$ represent the error vector of the PNN-MCD model on set R, where $e_{FL}(\mathbf{x}_s{}^R) = |\, c_s{}^{TR} - f_{FL}(\mathbf{x}_s{}^R)\,|$ and $e_{PNN\text{-}MCD}\,(\mathbf{x}_s{}^R) = |\, c_s{}^{TR} - f_{PNN\text{-}MCD}(\mathbf{x}_s{}^R)\,|$.

Similarly, let $\mathbf{e}_{FL}{}^Q = [e_{FL}(\mathbf{x}_1{}^Q), e_{FL}(\mathbf{x}_2{}^Q), \ldots, e_{FL}(\mathbf{x}_{NQ}{}^Q)]$ represent the error vector of the fuzzy logic model on set Q, and $\mathbf{e}_{PNN\text{-}MCD}{}^Q = [e_{PNN\text{-}MCD}\,(\mathbf{x}_1{}^Q), e_{PNN\text{-}MCD}\,(\mathbf{x}_2{}^Q), \ldots, e_{PNN\text{-}MCD}\,(\mathbf{x}_{NQ}{}^Q)]$ represent the error vector of the PNN-MCD model on set Q, where $e_{FL}(\mathbf{x}_s{}^Q) = |\, c_s{}^{TQ} - f_{FL}(\mathbf{x}_s{}^Q)\,|$ and $e_{PNN\text{-}MCD}\,(\mathbf{x}_s{}^Q) = |\, c_s{}^{TQ} - f_{PNN\text{-}MCD}(\mathbf{x}_s{}^Q)\,|$.

Table 7 presents a performance comparison of the fuzzy logic and PNN-MCD models by calculating *AR* values and *p*-values through the application of both paired *t*-tests and independent *t*-tests on randomly selected sets R and Q. The testing experiment was repeated 100 times, and the results in the table represent the average results.

An analysis of the *AR* values obtained from the fuzzy logic model on the R and Q sets ($AR\,|_{R\text{-set}} = 0.823$, $AR\,|_{Q\text{-set}} = 0.818$) shows that they are very close to the *AR* value from the basic testing on the entire T set, which is 0.820. Additionally, the PNN-MCD model demonstrates perfect classification accuracy on both subsets R and Q, indicating a high level of consistency in classification accuracy for both models across different data subsets.

Applying the paired *t*-test to the two different sets R and Q for the fuzzy logic model yielded a *p*-value of 0.503. This value is significantly higher than the 0.05 significance level, indicating that the fuzzy logic model is stable and robust across different samples from the same data distribution. The PNN-MCD model behaves as an ideal classifier within the operational range of $8619 \leq \sigma \leq 244109$. Due to the absence of variance in the error vectors, a paired *t*-test could not be applied to the PNN-MCD model.

The independent *t*-test applied to compare the fuzzy logic model and the PNN-MCD model resulted in a *p*-value of 0.056. Although this value is close to the significance threshold of 0.05, it indicates that the difference in classification accuracy in favor of the PNN-MCD model is not statistically insignificant. This suggests that the observed

difference is unlikely to be due to random variation in the combinations of input variable values used by the models.

Table 7. Comparison of statistical test results for the fuzzy logic and PNN-MCD models, presenting model accuracy (*AR*) and *t*-test *p*-values based on testing with the R and Q sets.

Fuzzy Logic Model		**PNN-MCD Model**	
R set	Q set	R set	Q set
$AR = 0.823$	$AR = 0.818$	$AR = 1$	$AR = 1$
$p = t\text{-test}(\mathbf{e}_{\text{FL}}{}^{\text{R}}, \mathbf{e}_{\text{FL}}{}^{\text{Q}})$ $p = 0.503$		$p = t\text{-test}(\mathbf{e}_{\text{PNN-MCD}}{}^{\text{R}}, \mathbf{e}_{\text{PNN-MCD}}{}^{\text{Q}})$ The PNN-MCD model behaves as an ideal classifier within the operational range of $8619 \leq \sigma \leq 244{,}109$. Due to the absence of variance in the error vectors, the *t*-test is not applicable.	
$T = P \cup Q$		$T = P \cup Q$	
$AR = 0.820$		$AR = 1$	
$p = t\text{-test}(\mathbf{e}_{\text{FL}}{}^{\text{R}}, \mathbf{e}_{\text{PNN-MCD}}{}^{\text{R}})$ $p = 0.056$			

6. Conclusions

As numerous successful intelligent applications have already been briefly outlined, agent-based technology plays a crucial role in managing e-services. The models proposed in this paper introduce an intelligent agent-based management system designed to assist CERT in identifying potential attack types via an e-form, without requiring prior knowledge of harmful code. In this study, we used the operating system and RAM processes on a virtual machine to test malware behavior. Typically, many antivirus companies share virus signatures to detect and eliminate malicious code. In developing our intelligent e-service, specific characteristics of malicious code in its various forms are employed to train agents to respond to cyber-attacks. The collected dataset is used to compare the infected malware with non-infected malware processes on the user's computer.

Fuzzy logic and neural networks are complementary techniques in many intelligent prediction systems. In this research, we applied both of these methods to detect malicious code. To develop an ideal malware detection e-service, we trained the intelligent system using three variations of fuzzy partition methods to establish linguistic labels for malicious feature values. These linguistic labels were then used to evaluate the most effective rules for the fuzzy-based model, which is one of the two implemented agent-based models. The second model was based on a probabilistic neural network.

Overall, the probabilistic neural network demonstrated superior clustering performance, achieving an average accuracy of 100%. The fuzzy logic model followed closely, with an average accuracy of 82%. Based on the statistical testing of both models using *t*-test methods, it can be concluded that they exhibit stable performance and consistent classification across different subsets of the same data distribution. Moreover, the results confirm that the superior accuracy of the PNN-MCD model, compared to the fuzzy logic model, is not the result of random sample selection. The study's results suggest that the agent-based model, identified through numerous classification methods, significantly contributes to the effectiveness of agent-based management. In other words, fuzzy-based methods outperformed direct imitation rules when engaging artificial intelligence in intrusion detection systems (IDS).

We believe that the automated process of adjusting class label names to train classification models on unlabeled data can be further exploited in the domains of intelligent e-systems, automatic identification technologies, and artificial agent-based systems. This approach has significant potential for enhancing cybersecurity solutions by detecting and preventing attacks.

Author Contributions: Conceptualization, K.K. and O.P.-R.; methodology, Z.S. and O.P.-R.; software, A.S., Z.S. and O.P.-R.; validation, B.P., P.Č. and M.J.; formal analysis, P.Č.; investigation, K.K.; resources, M.J.; data curation, Z.S.; writing—original draft preparation, K.K., Z.S., O.P.-R.; writing—review and editing, K.K., Z.S. and O.P.-R.; visualization, P.Č.; supervision, B.P.; project administration, A.S.; funding acquisition, M.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Science, Technological Development and Innovation of the Republic of Serbia grant number 451-03-65/2024-03/200102.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to legal reasons.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Q.; Lu, J.; Jin, Y. Artificial intelligence in recommender systems. *Complex Intell. Syst.* **2021**, *7*, 439–457. [CrossRef]
2. Pivk, A.; Gams, M. Intelligent Agents in E-commerce. *Electrotech. Rev.* **2000**, *67*, 251–260.
3. Rizmal, I.; Radunović, V.; Krivokapić, Đ. *Guide through Information Security in the Republic of Serbia*; Centre for EuroAtlantic Studies—CEAS OSCE Mission to Serbia: Belgrade, Serbia, 2018; p. 19.
4. Law on Information Security ("Official Gazette of RS" No 6/16). Available online: http://www.parlament.gov.rs/upload/archive/files/cir/pdf/zakoni/2016/3515-15.pdf (accessed on 26 September 2021).
5. Virustotal. Available online: https://www.virustotal.com/ (accessed on 22 March 2022).
6. Lin, C.-J.; Huang, M.-S.; Lee, C.-L. Malware Classification Using Convolutional Fuzzy Neural Networks Based on Feature Fusion and the Taguchi Method. *Appl. Sci.* **2022**, *12*, 12937. [CrossRef]
7. Zhang, D.; Wu, X.; He, E.; Guo, X.; Yang, X.; Li, R.; Li, H. Android Malware Detection Based on Hypergraph Neural Networks. *Appl. Sci.* **2023**, *13*, 12629. [CrossRef]
8. Atacak, İ. An Ensemble Approach Based on Fuzzy Logic Using Machine Learning Classifiers for Android Malware Detection. *Appl. Sci.* **2023**, *13*, 1484. [CrossRef]
9. Specht, D.F. Probabilistic neural networks. *Neural Netw.* **1990**, *3*, 109–118. [CrossRef]
10. Chapman, I.M.; Leblanc, S.P.; Partington, A. Taxonomy of cyber-attacks and simulation of their effects. In Proceedings of the 2011 Military Modeling & Simulation Symposium, Boston, MA, USA, 3–7 April 2011; Society for Computer Simulation International: San Diego, CA, USA, 2011; pp. 73–80.
11. Dener, M.; Ok, G.; Orman, A. Malware Detection Using Memory Analysis Data in Big Data Environment. *Appl. Sci.* **2022**, *12*, 8604. [CrossRef]
12. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R.P.; Tang, J.; Liu, H. Feature selection: A data perspective. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–45. [CrossRef]
13. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [CrossRef]
14. Kuk, K.; Stanojević, A.; Jovanović, M.; Nedeljković, S. Intelligent E-Service for Detecting Malicious Code Based Agent Technology. In Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics (WIMS'18), Novi Sad, Serbia, 25–27 June 2018; ACM: New York, NY, USA, 2018; Volume 44.
15. Schuster, A. The impact of Microsoft Windows pool allocation strategies on memory forensics. *Digit. Investig.* **2008**, *5*, S58–S64. [CrossRef]
16. Ross, T.J. *Fuzzy Logic with Engineering Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
17. Casillas, J.; Cordón, O.; Herrera, F.; Magdalena, L. Accuracy improvements to find the balance interpretability-accuracy in linguistic fuzzy modeling: An overview. In *Accuracy Improvements in Linguistic Fuzzy Modeling*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 3–24.
18. Behadada, O.; Trovati, M.; Chikh, M.; Bessis, N. Big data-based extraction of fuzzy partition rules for heart arrhythmia detection: A semi-automated approach. *Concurr. Computat. Pract. Exp.* **2016**, *28*, 360–373. [CrossRef]
19. Guillaume, S.; Magdalena, L. Expert guided integration of induced knowledge into a fuzzy knowledge base. In *Soft Computing*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 10, pp. 773–784.
20. Grégory, S.; Olivier, P.; Toan, D. On Dissimilarity Measures at the Fuzzy Partition Level. In Proceedings of the 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Cadiz, Spain, 11–15 June 2018; Volume 17, pp. 301–312.
21. Yongli, L.; Xiaoyang, Z.; Jingli, C.; Hao, C.A. Validity Index for Fuzzy Clustering Based on Bipartite Modularity. *J. Electr. Comput. Eng.* **2019**, *1*, 2719617.
22. Chen, M.Y. Establishing interpretable fuzzy models from numerical data. In Proceedings of the IEEE 4th World Congress on Intelligent Control and Automation, Shanghai, China, 10–14 June 2002; pp. 1857–1861.
23. Chowdhury, S.; Rahul, K. Evaluation of approximate fuzzy membership function using linguistic input-an approached based on cubic spline. *JINAV J. Inf. Vis.* **2020**, *1*, 53–59. [CrossRef]

24. Berkachy, R.; Donzé, L. Linguistic questionnaire evaluation: An application of the signed distance defuzzification method on different fuzzy numbers. The impact on the skewness of the output distributions. *Int. J. Fuzzy Syst. Adv. Appl.* **2016**, *3*, 12–19.
25. Dubois, D. The role of fuzzy sets in decision sciences: Old techniques and new directions. *Fuzzy Sets Syst.* **2011**, *184*, 3–28. [CrossRef]
26. Torres-García, A.A.; Garcia, C.A.R.; Villasenor-Pineda, L.; Mendoza-Montoya, O. (Eds.) *Biosignal Processing and Classification Using Computational Learning and Intelligence: Principles, Algorithms, and Applications*; Academic Press: Cambridge, MA, USA, 2021.
27. Belabed, I.; Alaoui, M.T.; El Miloud, J.; Belabed, A. Association rules algorithms for data mining process based on multi agent system. In Proceedings of the 2019 International Conference on Machine Learning for Networking, Paris, France, 3–5 December 2019; Springer: Cham, Switzerland, 2019; pp. 431–443.
28. De Meo, P.; Quattrone, G.; Ursino, D. Using intelligent agents in e-government for supporting decision making about service proposals. In Proceedings of the 2006 International Symposium on Methodologies for Intelligent Systems, Bari, Italy, 27–29 September 2006; Springer: Berlin/Heidelberg, 2006; pp. 147–156.
29. Asabe, S.A.; Garba, E.J.; Ahmadu, A.D. Comparative Evaluation of Academic Performance in Waikato Environment for Knowledge Analysis Using Multiple Classification Algorithms. *Int. Res. J. Innov. Eng. Technol.* **2021**, *5*, 73–80.
30. Agrawal, R.; Imielinski, T.; Swami, A. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 26–28 May 1993; pp. 207–216.
31. Hipp, J.; Güntzer, U.; Nakhaeizadeh, G. *Algorithms for Association Rule Mining—A General Survey and Comparison*; ACM SIGKDD Explorations Newsletter: New York, NY, USA, 2000; Volume 2, pp. 58–64.
32. Mao, K.Z.; Tan, K.C.; Ser, W. Probabilistic Neural-Network Structure Determination for Pattern Classification. *IEEE Trans. Neural Netw.* **2000**, *11*, 1009–1016. [CrossRef]
33. El Emary, I.M.M.; Ramakrishnan, S. On the Application of Various Probabilistic Neural Networks in Solving Different Pattern Classification Problems. *World Appl. Sci. J. IDOSI Publ.* **2008**, *4*, 772–780.
34. Stankovic, Z.; Doncov, N.; Milovanovic, B.; Milovanovic, I. Efficient DoA Tracking of Variable Number of Moving Stochastic EM Sources in Far-Field Using PNN-MLP Model. *Int. J. Antennas Propag.* **2015**, *2015*, 542614. [CrossRef]
35. Adeli, H.; Panakkat, A. A probabilistic neural network for earthquake magnitude prediction. *Neural Netw.* **2009**, *22*, 1018–1024. [CrossRef]
36. Wang, Z.; Jiang, M.; Hu, Y.; Li, H. An Incremental Learning Method Based on Probabilistic Neural Networks and Adjustable Fuzzy Clustering for Human Activity Recognition by Using Wearable Sensors. *IEEE Trans. Inf. Technol. Biomed.* **2012**, *16*, 691–699. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.