The distributivity law as a tool of k-NN classifiers' aggregation: mining a cyber-attack data set

1st Ewa Rak
University of Rzeszów
Rzeszów, Poland
ewarak@ur.edu.pl

2nd Jan G. Bazan
University of Rzeszów
Rzeszów, Poland
bazan@ur.edu.pl

3rd Adam Szczur
University of Rzeszów
Rzeszów, Poland
adamszczur8@gmail.com

4th Wojciech Rząsa
University of Rzeszów
Rzeszów, Poland
wrzasa@ur.edu.pl

***Abstract*—This contribution proposed a novel approach for an ensemble method to increase classification accuracy and at the same time minimizing ensemble classifiers by applying the distributivity law which will aggregate the classifiers accordingly. Ensemble methods have been introduced as a useful and effective solution to improve the performance of the classification. Despite having the ability of producing the highest classification accuracy, ensemble methods have suffered significantly from their large volume of base classifiers. Nevertheless, we could overcome this problem by combining some of the classifiers. We employ here the classical version of the k Nearest Neighbor classifiers (k-NN classifiers). Moreover, this method requires the use of some suitable aggregation operators for which either the distributivity law or one of its respective inequalities occurs. A good example of such aggregations were average functions and triangular norms and conorms. The paper includes primarily the results of experiments performed on the cyber attacks in network dataset obtained from the machine learning repository UCI.**

***Index Terms*—Ensemble method; k-NN method; Distributivity law; Aggregation functions; Accuracy.**

I. Introduction

Machine learning is one of the most promising approaches to address difficult decision problems. The general idea is very simple: instead of modeling a solution explicitly, a domain expert provides example data that demonstrate the desired behavior on representative problem instances. A suitable machine learning algorithm is then trained on these examples to reproduce the expert's solutions as well as possible and generalize it to new, unseen data. Ensemble methods or multiple classifiers are known as learning algorithms that train a set of classifiers and combine them to achieve the best prediction accuracy [1]. Previous works have shown that combining the predictions of a collection of classifiers can be an effective strategy to improve generalization performance, such as bagging [2], boosting [3], stacking [4], ensemble selection [5] and hybrid intelligent system [6]. The most fundamental concepts of ensemble methods consist of two main stages which is the production of multiple base classifier models and their combination via aggregation.

Aggregation functions proved to be an effective tool in many application areas [7], among which can be distinguished information retrieval [8]. It simply refers to the calculations performed on a data set to get a single number that accurately represents the underlying data. Actually, there are also many approaches to use domain knowledge and improve the quality of data mining models (see e.g., [9]-[13]). Thus, the use of aggregation functions can be treated as a way to use domain knowledge to improve the quality of classifiers.

Classification and cyber attack detection is a major challenge for web and network security. The increasing data traffic in network and web invites multiple cyber attack. The dynamic nature and large number of attributes of cyber data faced a problem of cyber attack detection and prevention. Data mining algorithms that classify the attacks must do it accurately. They must reduce unclassified attack to improve accuracy.

This study accordingly addresses the issue of aggregating the k-NN classifiers, through the use of distributivity law and preserving the relatively high classification accuracy. In general, the accuracy parameter (ACC) is the ratio of number of correct predictions to the total number of input samples. It usually works well if there are comparable number of samples belonging to each class (balanced decision classes). We will show that the new method of classification presented in this paper increases its quality compared to the accuracy obtained on raw data.

The paper is organized as follows. In Section 2, notions connected with aggregation functions and distributivity between them are recalled. Section 3 discusses the k Nearest Neighbor algorithm. In Section 4, details concerning the used Data Set are presented. Section 5 describes the experimental setting and results. Finally, Section 6 summarizes this work.

II. Distributivity law of aggregation functions

Mathematical models based on human thinking, information processing and decision making require continuous improvement. Among the key issues is to tackle information overloading we can distinguish the development of the process known as data aggregation, which plays an important role in many areas of human interest. In the most general sense, the aggregation process is a synthesis of many numerical data to a single value, in some ways, a representative for all of them. This type of projection methods of multidimensional space input data to one dimension is usually carried out by the so-called aggregation functions (also known as aggregation operators or aggregation operations). Although the idea of aggregating information into a representative output is quite ancient, the mathematical study of aggregation functions has been formalized quite more recently, and since then have been extensively investigated and most of the known results in this topic have been compiled in several books (see, e.g., [14], [15] or [16]), that provide details of many aggregation methods.

The formal definition of an aggregation function in a binary case is the following.
Let $I = [a, b]$ be a subinterval (scale) of a extended real line. It is a matter of rescaling to fix $I = [0, 1]$ as more often applicable.

Definition 1 (cf. [17])**.** A binary aggregation function is a mapping $A : [0, 1]^2 \to [0, 1]$ such that
$A1)$ $A(0, 0) = 0$ and $A(1, 1) = 1$ (boundary conditions);
$A2)$ A is increasing in both variables i.e. $A(X, Y) \le A(Z, T)$ if $(X, Y) \le (Z, T)$, $X, Y, Z, T \in [0, 1]$.

There are a huge number of aggregation functions which are grouped into different families such as means, triangular operations, copulas, Choquet and Sugeno integrals, uninorms, nullnorms and many others (about 100 families). Below are examples of families of aggregation functions, which can easily be applied to the problem of classification.

Definition 2 (cf. [7])**.** A mean (or averaging function) M is an idempotent aggregation function, i.e. $M(X, X) = X$ for all $X \in [0, 1]$.

Definition 3 (cf. [18], p. 4,11)**.** A triangular norm T (triangular conorm S) is a commutative and associative aggregation function having a neutral element $e = 1$ ($e = 0$).

TABLE I
EXAMPLES OF BASIC MEANS, T-NORMS AND T-CONORMS (SEE [19]).

Mean	
$M_\wedge(X, Y) = \min(X, Y)$	$M_\vee(X, Y) = \max(X, Y)$
$M_A(X, Y) = \frac{X+Y}{2}$	$M_G(X, Y) = \sqrt{XY}$
$M_H(X, Y) = \begin{cases} 0, & X = Y = 0 \\ \frac{2XY}{X+Y}, & elsewhere \end{cases}$	$M_P(X, Y) = \sqrt{\frac{X^2+Y^2}{2}}$
T-norm	**T-conorm**
$T_M(X, Y) = \min(X, Y)$	$S_M(X, Y) = \max(X, Y)$
$T_P(X, Y) = X \cdot Y$	$S_P(X, Y) = X + Y - X \cdot Y$
$T_L(X, Y) = \max(X + Y - 1, 0)$	$S_L(X, Y)) = \min(X + Y, 1)$

Distributivity of multiplication with respect to the addition occurs naturally in the arithmetic of real numbers, in vector and matrix calculus. In general, it specifies the relationship between two binary functions including aggregation functions.

Definition 4 (cf. [20], p. 318)**.** Let F and G be some binary functions in the non-empty set U, where F is symmetric. We say that F is distributive over G if for all $X, Y, Z \in U$ the following equality is fulfilled

$$F(X, G(Y, Z)) = G(F(X, Y), F(X, Z)). \qquad (1)$$

If in Definition 4 an equality is replaced by inequalities " $\le$ " or " $\ge$ ", respectively, then we say that
F is subdistributive with respect to G if

$$F(X, G(Y, Z)) \le G(F(X, Y), F(X, Z)), \qquad (2)$$

F is superdistributive with respect to G if

$$F(X, G(Y, Z)) \ge G(F(X, Y), F(X, Z)). \qquad (3)$$

At present, many studies are dealing with the distributivity law for different operators defined on the unit interval that are essential in decision making and utility theories [21], [22], fuzzy logic theory or in image processing [23], [24].

Due to the demand for these practical applications, discussion on the distributivity law between various functions, including aggregation functions, have revived (see e.g., [25], [26], [27],[19], [28]). [29] characterized, among others, solutions of the distributivity equation for averaging and quasilinear functions.

The lack of distributivity is a big problem in any algebraic transformations, and therefore also in computer modeling (see e.g., [30]). In general, aggregations are not distributive from each other, and still less mutually distributive. In our opinion the best is to illustrate this problem on the example of means and triangular norms and conorms (Table I). As a result, between the right and the left side of the distributivity law (DL) there may occur four different relations $L \le P$, $L \ge P$, $L = P$ and $L \parallel P$ (both sides of (1) are incomparable), as was summarized in Table II.

TABLE II
DISTRIBUTIVITY OF T-NORMS, T-CONORMS AND MEANS (SEE [19]).

$\mathbf{F \setminus G}$	$\mathbf{S_L}$	$\mathbf{S_P}$	$\mathbf{S_M}$	$\mathbf{M_P}$	$\mathbf{M_A}$	$\mathbf{M_G}$	$\mathbf{M_H}$	$\mathbf{T_M}$	$\mathbf{T_P}$	$\mathbf{T_L}$
$\mathbf{S_L}$	$\le$	$\parallel$	$=$	$\ge$	$\ge$	$\parallel$	$\parallel$	$=$	$\parallel$	$\parallel$
$\mathbf{S_P}$	$\le$	$\le$	$=$	$\ge$	$=$	$\le$	$\le$	$=$	$\ge$	$\ge$
$\mathbf{S_M}$	$\le$	$\le$	$=$	$\le$	$\le$	$\le$	$\le$	$=$	$\ge$	$\ge$
$\mathbf{M_P}$	$\le$	$\le$	$=$	$=$	$\le$	$\le$	$\le$	$=$	$\ge$	$\ge$
$\mathbf{M_A}$	$\le$	$\le$	$=$	$\ge$	$=$	$\le$	$\le$	$=$	$\ge$	$\ge$
$\mathbf{M_G}$	$\le$	$\le$	$=$	$\ge$	$\ge$	$=$	$\le$	$=$	$\ge$	$\parallel$
$\mathbf{M_H}$	$\le$	$\le$	$=$	$\ge$	$\ge$	$\ge$	$=$	$=$	$\parallel$	$\parallel$
$\mathbf{T_M}$	$\le$	$\le$	$=$	$\ge$	$\ge$	$\ge$	$\ge$	$=$	$\ge$	$\ge$
$\mathbf{T_P}$	$\le$	$\le$	$=$	$=$	$=$	$=$	$=$	$=$	$\ge$	$\ge$
$\mathbf{T_L}$	$\parallel$	$\parallel$	$=$	$\le$	$\le$	$\ge$	$\ge$	$=$	$\parallel$	$\ge$

III. K-NN ALGORITHM

Classification is a technique of data mining that is concerned about developing models that accurately predicts the classes of data object. After this is done, the developed model is then used to predict the class of objects whose class is not known. Data mining offers various techniques for classification such as k-NN, decision tree, SVM and rule based classification.

The k Nearest Neighbor (k-NN) classifier is one of the most widely used supervised classification methods [7]. In supervised learning problems, objects from a training set are preclassified into several categories or classes. To deal with such type of data we use the decision tables of the form $T = (U, A, dec)$ in Pawlak's sense [31], where A is a set of attributes or conditions (columns) in the data table, U is a set of objects (rows) in the data table and $dec \notin A$ is a distinguished attribute called decision attribute. In practice, the decision tables contain a description of a finite sample U_1 of objects from larger (even maybe infinite) universe U. Conditions are such attributes that their values are known for all objects from U, but decision is a function defined on the objects from the sample U_1 only. To construct the k-NN classifier the distance measure and the best number k of nearest neighbors must be chosen. As a distance the Euclidean or the "city" measure can be used. To select the best k it is necessary to calculate a probability of misclassification for $k = 1, 2, ..., n$, where n is a number of of objects in the training set, called also a learning or a reference set. The k which offers the minimum misclassification probability estimated from the learning set should be taken as optimal one.

In this technique when a new unclassified item x must be classified, the classifier searches the available training data and retrieves the k nearest data items (neighbors) to x according to a distance metric. Then, x is classified to the most common class indicated via a majority vote of the nearest neighbors. If more than one classes are most common (ties in voting), the major class is determined either randomly or by choosing the class of the nearest neighbor. The k-NN classifier is effective, easy to implement and has many applications. The classification cost depends on the size of the training set. In originally defined version of k-NN for each incoming new item x, the classifier has to compute distances between x and all training items. Unfortunately, cost of such algorithm may be very high and prohibitive for large data sets. However, nowadays more efficient versions of k-NN are known and used. For example, two of them, namely KDTree and BallTree, are implemented in Python programming language library scikit learn. See also [32] or references in that paper.

In order to test the performance of a classification model, its classification accuracy (ACC) is assessed, i.e.
Accuracy $= \frac{\textit{Number of correct predictions}}{\textit{Total number of predictions made}}$.

IV. Data set

In our deliberations we use KDDCup'99 of cyber attack in military network dataset (DARPA 1998 program) obtained from UC Irvine (UCI) Machine Learning Respository [33].
A variety of attacks incorporated in the dataset fall into following four major categories:
• **Denial of Service Attacks (DoS)**: A denial of service attack is an attack where the attacker constructs some computing or memory resource fully occupied or unavailable to manage legitimate requirements, or reject legitimate users right to use a machine.
• **User to Root Attacks (U2R)**: User to Root exploits are a category of exploits where the attacker initiate by accessing a normal user account on the system and take advantage of some susceptibility to achieve root access to the system.
• **Remote to User Attacks (R2L)**: A Remote to User attack takes place when an attacker who has the capability to send packets to a machine over a network but does not have an account on that machine, makes use of some vulnerability to achieve local access as a user of that machine.
• **Probes**: Probing is a category of attacks where an attacker examines a network to collect information or discover well-known vulnerabilities. These network investigations are reasonably valuable for an attacker who is staging an attack in future. An attacker who has a record, of which machines and services are accessible on a given network, can make use of this information to look for fragile points.

In KDDCup'99 dataset these four attack classes (DoS, U2R, R2L, and probing) are divided into 22 more particular attack classes as shown in Table III. The KDDCup'99 datasets are divided into two parts: the training dataset and the testing dataset. The testing dataset contains not only known attacks from the training data but also unknown attacks. Since 1999, KDDCup'99 has been the most wildly used data set for the evaluation of anomaly detection methods. DARPA'98 is about 4 gigabytes of compressed raw (binary) tcp-dump data of 7 weeks of network traffic, which can be processed into about 4 898 431 single connection records. For each connection consist 41 features marked as either normal or an attack, with exactly one particular attack type. 41 various quantitative (continuous data type) and qualitative (discrete data type) features were extracted among the 41 features, 34 features are numeric and 7 features are symbolic - omitted in our experiments.

TABLE III
Different types of attacks in KDDCup'99 dataset.

22 Attack Classes	Main Attack Classes	Cardinality of Classes
normal	–	972 781
smurf	DoS	2 807 886
neptune	DoS	1 072 017
satan	Probing	15 892
ipsweep	Probing	12 481
portsweep	Probing	10 413
nmap	Probing	2 316
back	DoS	2 203
warezclient	R2L	1 020
teardrop	DoS	979
pod	DoS	264
guess_passwd	R2L	53
butter_overflow	U2R	30
land	DoS	21
warezmaster	R2L	20
imap	R2L	12
rootkit	U2R	10
leadmodule	U2R	9
ftp_write	R2L	8
multihop	R2L	7
phf	R2L	4
perl	U2R	3
spy	R2L	2

To perform our experiments we sampled data as follows:

- Chose 10 largest classes of attack types, including normal (no attack) and 9 attacks classes (see Table III).
- Took 181216 records (data sample named E181216):
    - Took all 45304 objects from smallest decision classes (satan - 15892, ipsweep - 12481, portsweep - 10413, nmap -2316, back - 2203, warezclient - 1020, teardrop - 979);
    - Chose randomly 45304 objects from remaining classes (normal, smurf, neptune).

V. Aggregate (ensemble) method for k-NN classifiers

As mentioned in the Introduction many researchers have proposed various ensemble methods as learning algorithms in data mining to improve the classifiers performance and accuracy. There is no single ensemble methods that dominate classification technique. Most of the previous studies focus on the ensemble construction and ensemble combination in improving the accuracy and performance of classification. We add here also our approach to increase classification accuracy and at the same time minimizing ensemble classifiers by applying the distributivity law which will aggregate the classifiers accordingly (see the scheme in Fig. 1).

A. *Steps in the proposed classification approach*

Let d_0 be the decision class "normal" (no attacks) and $d_1, d_2, ..., d_9$ denote 9 types of network attacks, respectively. Moreover $\varepsilon \in (0,1)$ denote the threshold for classifying an object into one of the d_i classes. The use of ε parameter in experiments is described later in this chapter where the algorithm for the classification of the test object has been placed.

- Construct the classifiers as follows:
  (1.) The **P** classifier - used to pre-classify an object as to whether it belongs to decision class d_0 (label 0) or to some other decision class $\neg d_0 = \{d_1, d_2, ..., d_m\}$ (label 1);
  (2.) Collection of classifiers $\mathbf{C_i}$ $(i = 1, 2..., m)$ - used to distinguish objects with decision value d_i (label 1) from objects from other classes $\neg d_i = \{d_1, d_2, ..., d_m\} \setminus \{d_i\}$ (label 0) without the class d_0;
  (3.) Collection of classifiers $\mathbf{P_i}$ $(i = 1, 2..., m)$ - used to distinguish objects with decision value d_i (label 1) from objects of the class d_0 (label 0).

  The method used for $\mathbf{P}, \mathbf{P_i}, \mathbf{C_i}$ classifiers construction: k-NN (IBk) from WEKA API (see [34]) with Euclidean distance and $k = \sqrt{n}$, where n - number of objects (c.f. [35]).
- Determine the probability of membership to decision class with label "1" (weight for class labeled "1") for each instance. Next construct the weight table of classifiers $\mathbf{P}, \mathbf{C_i}, \mathbf{P_i}$ for the class labeled "1" (classifiers as columns, objects as rows, weights (probabilities) for class "1" as values).
- For a given distributivity law (1) specified in Table II compute the value of its left side L considering the obtained classifiers X, Y, Z in every possible combination of their settings i.e.
  $X = P, Y \in C_1, ..., C_9, Z \in P_1, ..., P_9;$
  $X = P, Y \in P_1, ..., P_9\ Z \in C_1, ..., C_9;$
  $X \in C_1, ..., C_9, Y = P, Z \in P_1, ..., P_9;$
  $X \in P_1, ..., P_9, Y = P, Z \in C_1, ..., C_9;$
  $X \in C_1, ..., C_9, Y \in P_1, ..., P_9, Z = P;$
  $X \in P_1, ..., P_9, Y \in C_1, ..., C_9, Z = P.$
  In the case of subdistributivity (2) and superdistributivity (3) calculate their both sides left L and right R.
- Fix the parameter $\varepsilon \in (0, 1)$.
- Determine the maximum for the previously calculated values of both left and right sides.
- For each tested object propose decision value as follows: If the maximum value of the left (right) side $(\max_{aggr})$ of distributivity (subdistributivity or superdistributivity) was obtained for the i-th decision (after aggregating the weights of the classifiers $\mathbf{P}, \mathbf{C_i}, \mathbf{P_i}$) and $\max_{aggr} > \varepsilon$, then propose the i-th decision value for the given object; else if $\max_{aggr} \leq \varepsilon$ then propose a neutral decision ("normal").

The above proposed algorithm has been optimized to improve the command classification accuracy.

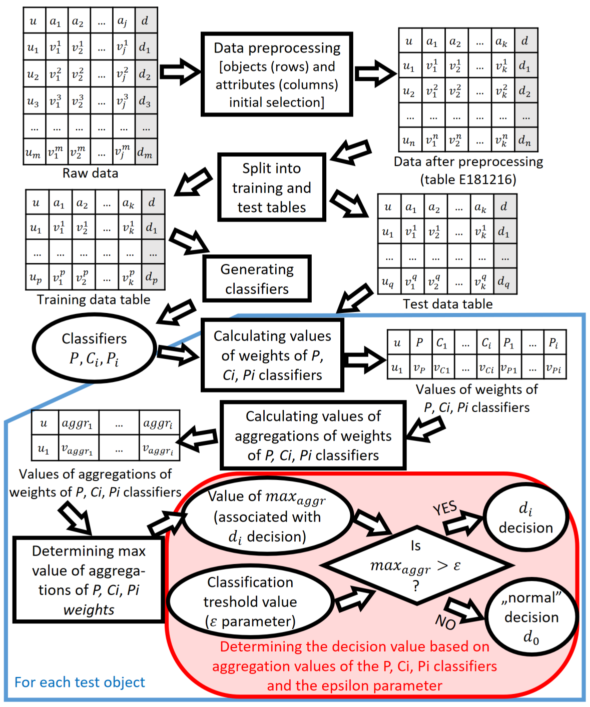
Raw data
Data preprocessing [objects (rows) and attributes (columns) initial selection]
Data after preprocessing (table E181216)
Split into training and test tables
Training data table
Test data table
Generating classifiers
Classifiers P, C_i, P_i
Calculating values of weights of P, Ci, Pi classifiers
Values of weights of P, Ci, Pi classifiers
Calculating values of aggregations of weights of P, Ci, Pi classifiers
Values of aggregations of weights of P, Ci, Pi classifiers
Determining max value of aggrega-tions of P, Ci, Pi weights
Value of max_{aggr} (associated with d_i decision)
Classification treshold value (ε parameter)
Is $max_{aggr} > \varepsilon$?
YES
NO
d_i decision
„normal" decision d_0
Determining the decision value based on aggregation values of the P, Ci, Pi classifiers and the epsilon parameter
For each test object

Fig. 1. The scheme of the proposed approach.

B. *Experiments*

An algorithm was implemented and tested in Java programming language (using Weka API library for $\mathbf{P}, \mathbf{C_i}, \mathbf{P_i}$ classifiers construction cf. [34]).
It uses the distributivity law and its corresponding inequalities specified in Table II. The examples selected for the analysis are summarized in the Table IV, where D1-D5 specify the distributivity law (1) and SubD1-SubD2 and SupD1-SupD2, respectively subdistributivity and superdistributivity for suitable aggregation functions.

For testing quality of classifiers we applied 10-times train&test technique. Each time data E181216 were divided into 2 equal subsets for training and testing, respectively. When splitting data, the proportions of decision classes were maintained.

TABLE IV
EXAMPLES OF EXAMINED DISTRIBUTIVITY EQUATION (1) AND INEQUALITIES (2), (3) INCLUDED IN TABLE II.

(1)	D1	$L = T_P(P, \min(C_i, P_i)) = \min(T_P(P, C_i), T_P(P, P_i)) = R$
	D2	$L = T_P(P, M_A(C_i, P_i)) = M_A(T_P(P, C_i), T_P(P, P_i)) = R$
	D3	$L = M_A(P, M_A(C_i, P_i)) = M_A(M_A(P, C_i), M_A(P, P_i)) = R$
	D4	$L = T_P(P, M_H(C_i, P_i)) = M_H(T_P(P, C_i), T_P(P, P_i)) = R$
	D5	$L = M_P(P, M_P(C_i, P_i)) = M_P(M_P(P, C_i), M_P(P, P_i)) = R$
(2)	SubD1	$L = \max(P, M_A(C_i, P_i)) \leq M_A(\max(P, C_i), \max(P, P_i)) = R$
	SubD2	$L = S_P(P, M_H(C_i, P_i)) \leq M_H(S_P(P, C_i), S_P(P, P_i)) = R$
(3)	SupD1	$L = \min(P, M_A(C_i, P_i)) \geq M_A(\min(P, C_i), \min(P, P_i)) = R$
	SupD2	$L = M_A(P, T_P(C_i, P_i)) \geq T_P(M_A(P, C_i), M_A(P, P_i)) = R$

To asses the quality of the proposed method the accuracy (ACC) was used. The results of ACC measurements on the original raw data and newly created data with the fixed $\varepsilon \in (0, 1)$ are presented in the tables located in the Appendix Section at the end of this contribution.

Due to the large size of result tables and limited space of this paper, for SubD1, SubD2, SupD1 and SupD2 results for $\varepsilon \in (0, 0.4]$ are presented (see Appendix). For larger ε values and these aggregations, the following dependencies can be observed:

- if ε increases, the overall result gradually decreases;
- for classes *smurf* and *neptune*, the results are the same regardless of the ε value;
- for other types of attacks, the value of ACC decreases if ε increases;
- in the case of the *normal* class, the increase of ε parameter value improves the results, and the best results are obtained for $\varepsilon = 0.9$ (for SubD1 $L = R = 0.997$, for SubD2 $L = R = 0.991$, for SupD1 $L = R = 0.998$ and for SupD2 $L = R = 0.999$).

C. Discussion of the results

In general, we can observe that the obtained results justify our approach of using the idea of distributivity of aggregation functions in classification method. For each of the tested distributivity equation D1-D5, the new classification method gives better ACC (overall) results:

- for D1 we have the best result with $\varepsilon = 0.1$
- for D2 the best result is obtained at $\varepsilon = 0.2$ or $\varepsilon = 0.3$
- for D3 the best result is obtained at $\varepsilon = 0.5$
- for D4 the best result is obtained at $\varepsilon = 0.2$
- for D5 we have the best result with $\varepsilon = 0.6$

In the case of subdistributivity (SubD1-SubD2), unfortunately, our method gives worse results compared to the k-NN classifier calculated on raw data.
In the case of the tested superdistributivity (SupD1-SupD2):

- for SupD1 we get the best total score when $\varepsilon = 0.3$ (the left side L is equal and the right side R is better than ACC on the raw data) or $\varepsilon = 0.4$ (both left L and right R are better than ACC on the raw data)
- for SupD2 the best result is obtained when $\varepsilon = 0.2$ (the right side R of (3) gives better ACC than raw data) or $\varepsilon = 0.3$ (both left L and right R side gives the best results) or $\varepsilon = 0.4$ (the left side L is the best).

In addition, for all tested distributivity D1-D5 all of the attacks are very vell recognized, though mostly for $\varepsilon = 0.1$ and $\varepsilon = 0.2$. In the case of superdistributivity (SupD1-SupD2) respective attacks are also better recognized when $\varepsilon = 0.1$, except that:

- for SupD1, some attacks are better recognized using the left side L (e.g., *warezclient*, *teardrop*), some attacks by the right side R (*satan*, *nmap*), and some by both sides L and R (*neptune*, *ipsweep*, *portsweep*),
- for SupD2, the best recognition of attacks is when using the left side L of (3), but sometimes right side R gives the same results.

Moreover, the potential power of the distributive law (1) is that its right side needs to perform three operations, whereas the left side needs only two. Thus, the distributive law gives us a "fast algorithm" for computing and it can be vastly generalized (see [36]).

The donator of data set KDDCup'99, i.e. Lincoln Labs said that "they set up an environment to simulate a typical U.S. Air Force LAN". They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks. If so, the data may be treated as a sample of a population of all attacks on U.S. Air Force LAN. That's why we decided to discover statistically significant corollaries concerning the outputs of experiments (i.e. corollaries about all records of KDDCup'99 data set with chosen 10 kinds of cyber-attacks). The presented below corollaries, received by employing the Wilcoxon matched pairs tests. We used the test implemented in STATISTICA program [37].

For each studied approach from among D1-D5, SubD1-SubD2, SupD1-SupD2 we took into consideration 2 values of epsilon which gave the best overall outputs. It is done to maximally reduce number of multiple comparisons in proceeded statistical tests. We tested if accuracy of recognizing 9 kinds of cyber-attacks and normal activity are the same for raw data and for studied approaches - they were our null hypothesis. We present outputs of performed statistical tests for p values less than 0.1. Raw data approach works worse than: D1 for $\varepsilon = 0.2$ with $p = 0.059$; D2 for $\varepsilon = 0.3$ with $p = 0.059$; D3 for $\varepsilon = 0.5$ with $p = 0.093$ and for $\varepsilon = 0.6$ with $p = 0.047$; D4 for $\varepsilon = 0.2$ and $\varepsilon = 0.3$ with $p = 0.059$; D5 for $\varepsilon = 0.6$ with $p = 0.092$ and for $\varepsilon = 0.7$ with $p = 0.080$.

VI. CONCLUSIONS

Ensembles are well established as a method for obtaining highly accurate classifiers by combining less accurate ones.

The efficiency of the proposed approach has been proven by comparing the classification accuracy obtained using the proposed method with the classical k-NN algorithm based on the raw dataset. With the appropriate value of the ε parameter (Section V-C), we obtained results up to 37.8% better (than for raw data) for individual decision classes (e.g., in the Table D3 and Table D5 for $\varepsilon = 0.1$ and decision class "*back*" - 37.8%, in the Table D3 and Table D5 for $\varepsilon = 0.1$ and decision class "*warezckient*" - 10-10.2%). By selecting the ε parameter we can adjust the sensitivity and specificity of the classifier, and thereby make the proposed method will better detect attacks or lack of them ”*normal*”.

It is worth adding that the construction method of a complex classifier presented in this work can be used not only for cyberattack data. These data should be treated only as an example for which the proposed approach to the construction of classifiers can be used. Let us remind also that this data has the property of a hierarchical structure of decision classes, i.e. from a general point of view there is a division of objects into two decision classes: normal and attack. However, on the lower level of detail, the decision class on attacks is divided into 9 subclasses, corresponding to 9 types of attacks. This structure can have decision classes for many important applications related to the construction of classifiers. A very similar structure is data relating to fraud, e.g., credit card (fraud detection). Also, many medical data have a similar structure. For example, in the prediction of coronary heart disease, we generally divide patients into two classes, those who do not have coronary artery stenosis, and those who have such stenosis. However, more specifically, patients with coronary artery stenosis are divided into, for example, those with small, medium and large stenoses (see e.g., [10], [12], [38]). The presented approach can be easily adapted to the decision-making problems mentioned above, which we plan to do as a part of further research.

Acknowledgment

This work was partially supported by the Centre for Innovation and Transfer of Natural Sciences and Engineering Knowledge of University of Rzeszów, Poland, the project RPPK.01.03.00-18-001/10.

The presented work is also partly a continuation of the research carried out under the grant of the Polish National Science Centre: DEC-2013/09/B/ST6/01568.

References

[1] T. G. Dietterich, ”Ensemble methods in machine learning”, Multiple classifier systems, Springer Berlin Heidelberg, pp. 1-15, 2000.
[2] L. Breiman, ”Bagging predictors”, Mach. Learn., 24, pp. 123-140, 1996.
[3] Y. Freund and R. E. Schapire, ”Experiments with a new boosting algorithm”, ICML, 96, pp. 148-156, 1996.
[4] L. Breiman, ”Stacked regressions”, Mach. Learn., 24, pp. 49-64, 1996.
[5] J. J. Rodriguez, L. I. Kuncheva and C. J. Alonso, ”Rotation forest: A new classifier ensemble method”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28, pp. 1619-1630, 2006.
[6] R. Caruana, A., Niculescu-Mizil, G., Crew and A. Ksikes, ”Ensemble selection from libraries of models”, Proceedings of the twenty-first international conference on Machine learning ACM, pp. 18, 2004.
[7] G. Beliakov, H. Bustince and T. Calvo, ”A Practical Guide to Averaging Functions”, Stud. Fuzziness Soft Comput., vol. 329, Springer, Cham, 2016.
[8] S. Marrara, G. Pasi and M. Viviani,” Aggregation operators in information retrieval”, Fuzzy Sets Syst., 324, pp. 3-19, 2017.
[9] J. G. Bazan, S. Burgewa-Czuma and A. Jankowski, ”A domain knowledge as a tool for improving classifiers”, Fudam. Inform., 127, pp. 495-511, 2013.
[10] J. G. Bazan, S. Bazan-Socha, S. Buregwa-Czuma, L. Dydo, W. Rzasa and A. Skowron, ”A classifier based on a decision tree with verifying cuts”, Fundam. Inform., 143, pp. 1-18, 2016.
[11] J. G. Bazan, A. Szczur, A. Skowron, M. Rzepko, P. Król, W. Bajorek and W. Czarny, ”A Classifier Based on a Decision Tree with Temporal Cuts”, Fundam. Inform., 165, pp. 263-281, 2019.
[12] J. G. Bazan, S. Bazan-Socha, M. Ochab, S. Buregwa-Czuma, T. Nowakowski and M. Woźniak, ”Effective construction of classifiers with the k-NN method supported by a concept ontology”, Knowl Inf Syst., https://doi.org/10.1007/s10115-019-01391-w, 2019.
[13] U. Bentkowska, J. G. Bazan, W. Rząsa and L. Zaręba, ”Application of interval-valued aggregation to optimization problemof k-NN classifiers for missing values case”, Inf. Sci., 486, pp. 434-449, 2019.
[14] G. Beliakov, A. Pradera and T. Calvo, ”Aggregation Functions: A Guide for Practitioners”, Stud. Fuzziness Soft Comput., vol.221, Springer, Berlin, Heidelberg, 2007.
[15] T. Calvo, G. Mayor and R. Mesiar, ”Aggregation Operators: New Trends and Applications”, Stud. Fuzziness Soft Comput., vol. 97, Springer, Berlin, Heidelberg, 2002.
[16] M. Grabisch, J.L. Marichal, R. Mesiar and E. Pap, ”Aggregation Functions”, Encyclopedia of Mathematics and Its Applications, vol. 127, Cambridge University Press, New York, 2009.
[17] J. Dombi, ”Basic concepts for the theory of evaluation : The aggregative operator”, European J. Operat. Research, 10, pp. 282-293, 1982.
[18] E. P. Klement, R. Mesiar and E. Pap, ”Triangular norms”, Kluwer Acad. Publ., Dordrecht, 2000.
[19] J. Drewniak and E. Rak, ”Subdistributivity and superdistributivty of binary operations”, Fuzzy Sets Syst., 161, pp. 189-210, 2010.
[20] J. Aczél, ”Lectures on functional equations and their applications”, Academic press, New York, London, 1966.
[21] D. Dubois, E. Pap and H. Prade, ”Hybrid probabilistic-possibilistic mixtures and utility functions”, Preferences and Decisions under Incomplete Knowledge, Stud. Fuzziness Soft Comput., vol.51, Springer-Verlag, pp.51–73, 2000.
[22] D. Jočić and I. Štainer-Papuga, ”Some implications of the restricted distributivity of aggregation operators with absorbing elements for utility theory”, Fuzzy Sets Syst., 291, pp. 54-65, 2016.
[23] M. Galar, A. Jurío, C. López-Molina, D. Paternain, J. Sanz and H. Bustince, ”Aggregation functions to combine RGB color channels in stereo matching”, Opt. Express, 21, pp. 1247–1257, 2013.
[24] D. Paternain, J. Fernandez, H. Bustince, R. Mesiar and G. Beliakov, ”Construction of image reduction operators using averaging aggregation functions”, Fuzzy Sets Syst., 261, pp. 87–111, 2015.
[25] C. Alsina, E. Trillas and L. Valverde, ”On non-distributive logical connectives for fuzzy sets theory”, BUSEFAL, 3, pp. 18-29, 1980.
[26] M. Carbonell, M. Mas, J. Suñer and J. Torrens, ”On distributivity and modularity in De Morgan triplets”, Internat. J. Uncertainty, Fuzzines Knowledge-Based Syst., 4, pp. 351-368, 1996.
[27] J. Dombi, ”Properties of the fuzzy connectives in light of the general representations theorem”, Acta Cybernet., 7, pp. 313-321, 1986.
[28] J. Drewniak, P. Drygaś and E. Rak, ”Distributivity equations for uninorms and nullnorms”, Fuzzy Sets Syst., 159, pp. 1646-1657, 2008.
[29] T. Calvo, ”On some solutions of the distributivity equation”, Fuzzy Sets Syst., 104, pp. 85-96, 1999.
[30] W. E. Combs and J. E. Andrews, ”Combinatorial rule explosion eliminated by a fuzzy rule configuration”, IEEE Trans. Fuzzy Syst., 6, pp. 1–11, 1998.
[31] Z. Pawlak and A. Skowron, ”Rudiments of rough sets” Inf Sci., 177, pp. 28–40, 2007.
[32] W. Cherif, ”Optimization of K-NN algorithm by clustering and reliability coefficients: application to breast-cancer diagnosis”, Procedia Computer Science, 127 pp. 293–299, 2018.
[33] D. Dua and C. Graff, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019.
[34] E. Frank, M. A. Hall and I. H. Witten, The WEKA Workbench. Online Appendix for ”Data Mining: Practical Machine Learning Tools and Techniques”, Morgan Kaufmann, 4th Edition, 2016.
[35] A. B. Hassanat, M. A. Abbadi, G. A. Altarawneh and A. A. Alhasanat, ”Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach”, arXiv preprint arXiv:1409.0919, 2014.
[36] S. M. Aji and R. McEliece, ”The Generalized Distributive Law”, IEEE Trans. Inf. Theory, 46, pp. 325-342, 2000.
[37] StatSoft, Inc. STATISTICA (data analysis software system), version 13.
[38] S. Buregwa-Czuma, ”Methods of applying domain knowledge to improve the quality of classifiers (In Polish)”, PhD thesis, University of Silesia in Katowice, Faculty of Computer Science and Materials Science, Katowice, Poland, 2017.

APPENDIX

ACC	Raw data	**D1**								
		$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$
smurf	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000
neptune	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
satan	0.976 ± 0.001	**0.986** ± 0.001	0.980 ± 0.001	0.976 ± 0.001	0.969 ± 0.001	0.965 ± 0.001	0.961 ± 0.001	0.953 ± 0.001	0.942 ± 0.002	0.931 ± 0.003
ipsweep	0.964 ± 0.004	**0.972** ± 0.003	0.969 ± 0.004	0.966 ± 0.004	0.964 ± 0.004	0.962 ± 0.004	0.949 ± 0.007	0.931 ± 0.003	0.918 ± 0.005	0.896 ± 0.004
portsweep	0.945 ± 0.005	**0.955** ± 0.004	**0.955** ± 0.004	0.954 ± 0.004	0.954 ± 0.004	0.953 ± 0.004	0.936 ± 0.004	0.925 ± 0.003	0.919 ± 0.002	0.917 ± 0.002
nmap	0.825 ± 0.005	**0.858** ± 0.013	0.832 ± 0.007	0.832 ± 0.006	0.831 ± 0.006	0.826 ± 0.007	0.810 ± 0.007	0.788 ± 0.007	0.746 ± 0.017	0.655 ± 0.017
back	0.616 ± 0.017	**0.909** ± 0.018	0.710 ± 0.014	0.666 ± 0.011	0.585 ± 0.020	0.550 ± 0.010	0.537 ± 0.010	0.513 ± 0.012	0.506 ± 0.009	0.487 ± 0.011
warezclient	0.801 ± 0.084	**0.876** ± 0.008	0.875 ± 0.008	0.858 ± 0.037	0.617 ± 0.042	0.536 ± 0.032	0.465 ± 0.021	0.270 ± 0.045	0.032 ± 0.024	0.000 ± 0.000
teardrop	**0.959** ± 0.009	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.946 ± 0.007	0.923 ± 0.011	0.916 ± 0.011	0.912 ± 0.011
normal	0.988 ± 0.001	0.974 ± 0.001	0.986 ± 0.001	0.989 ± 0.001	0.992 ± 0.001	0.995 ± 0.001	0.997 ± 0.000	0.998 ± 0.001	0.998 ± 0.000	**0.999** ± 0.000
overall	0.981 ± 0.001	**0.984** ± 0.000	0.983 ± 0.001	0.983 ± 0.000	0.980 ± 0.001	0.980 ± 0.001	0.977 ± 0.000	0.973 ± 0.000	0.969 ± 0.001	0.965 ± 0.000

ACC	Raw data	**D2**								
		$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$
smurf	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000
neptune	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
satan	0.976 ± 0.001	**0.992** ± 0.000	0.984 ± 0.002	0.98 ± 0.001	0.973 ± 0.001	0.967 ± 0.001	0.964 ± 0.001	0.958 ± 0.001	0.950 ± 0.001	0.936 ± 0.002
ipsweep	0.964 ± 0.004	**0.972** ± 0.003	**0.972** ± 0.003	0.971 ± 0.003	0.966 ± 0.004	0.963 ± 0.004	0.960 ± 0.005	0.949 ± 0.005	0.938 ± 0.004	0.902 ± 0.006
portsweep	0.945 ± 0.005	**0.953** ± 0.004	**0.953** ± 0.003	**0.953** ± 0.003	0.952 ± 0.004	0.952 ± 0.004	0.952 ± 0.004	0.952 ± 0.004	0.935 ± 0.004	0.919 ± 0.003
nmap	0.825 ± 0.005	**0.858** ± 0.008	0.833 ± 0.007	0.832 ± 0.006	0.832 ± 0.006	0.832 ± 0.006	0.832 ± 0.006	0.825 ± 0.006	0.798 ± 0.007	0.730 ± 0.021
back	0.616 ± 0.017	**0.972** ± 0.005	0.885 ± 0.008	0.703 ± 0.015	0.653 ± 0.012	0.565 ± 0.011	0.548 ± 0.008	0.532 ± 0.009	0.510 ± 0.010	0.500 ± 0.010
warezclient	0.801 ± 0.084	**0.881** ± 0.009	0.877 ± 0.008	0.875 ± 0.008	0.765 ± 0.090	0.599 ± 0.015	0.514 ± 0.012	0.473 ± 0.018	0.320 ± 0.031	0.008 ± 0.008
teardrop	**0.959** ± 0.009	0.957 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.946 ± 0.007	0.916 ± 0.011
normal	0.988 ± 0.001	0.944 ± 0.002	0.974 ± 0.003	0.986 ± 0.001	0.990 ± 0.001	0.992 ± 0.001	0.996 ± 0.000	0.997 ± 0.000	0.998 ± 0.000	**0.999** ± 0.000
overall	0.981 ± 0.001	0.977 ± 0.001	**0.983** ± 0.001	**0.983** ± 0.000	0.982 ± 0.001	0.980 ± 0.000	0.980 ± 0.000	0.978 ± 0.001	0.974 ± 0.000	0.967 ± 0.001

ACC	Raw data	**D3**								
		$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$
smurf	0.999 ± 0.000	**1.000** ± 0.001	**1.000** ± 0.001	**1.000** ± 0.001	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000
neptune	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
satan	0.976 ± 0.001	**0.997** ± 0.000	0.994 ± 0.001	0.993 ± 0.001	0.990 ± 0.001	0.983 ± 0.002	0.976 ± 0.001	0.968 ± 0.001	0.962 ± 0.001	0.950 ± 0.002
ipsweep	0.964 ± 0.004	**0.972** ± 0.003	**0.972** ± 0.003	**0.972** ± 0.003	**0.972** ± 0.003	0.971 ± 0.003	0.968 ± 0.005	0.964 ± 0.004	0.958 ± 0.007	0.937 ± 0.004
portsweep	0.945 ± 0.005	**0.953** ± 0.004	**0.953** ± 0.004	**0.953** ± 0.004	**0.953** ± 0.003	**0.953** ± 0.003	0.952 ± 0.004	0.952 ± 0.004	0.952 ± 0.004	0.935 ± 0.004
nmap	0.825 ± 0.005	**0.870** ± 0.006	**0.870** ± 0.006	0.864 ± 0.010	0.840 ± 0.008	0.832 ± 0.006	0.832 ± 0.006	0.832 ± 0.006	0.831 ± 0.006	0.796 ± 0.007
back	0.616 ± 0.017	**0.994** ± 0.002	**0.994** ± 0.002	0.990 ± 0.003	0.969 ± 0.005	0.875 ± 0.013	0.683 ± 0.010	0.593 ± 0.024	0.544 ± 0.009	0.509 ± 0.009
warezclient	0.801 ± 0.084	**0.901** ± 0.009	0.895 ± 0.008	0.881 ± 0.009	0.878 ± 0.009	0.876 ± 0.008	0.861 ± 0.035	0.600 ± 0.014	0.504 ± 0.014	0.308 ± 0.030
teardrop	0.959 ± 0.009	**0.963** ± 0.006	0.962 ± 0.006	0.957 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.946 ± 0.007
normal	0.988 ± 0.001	0.007 ± 0.003	0.173 ± 0.033	0.928 ± 0.004	0.954 ± 0.003	0.980 ± 0.001	0.988 ± 0.001	0.992 ± 0.001	0.997 ± 0.000	**0.998** ± 0.000
overall	0.981 ± 0.001	0.744 ± 0.001	0.786 ± 0.008	0.974 ± 0.001	0.979 ± 0.001	**0.984** ± 0.000	0.983 ± 0.000	0.980 ± 0.001	0.979 ± 0.001	0.974 ± 0.000

ACC	Raw data	**D4**								
		$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$
smurf	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000
neptune	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
satan	0.976 ± 0.001	**0.989** ± 0.001	0.982 ± 0.002	0.979 ± 0.001	0.972 ± 0.001	0.967 ± 0.001	0.964 ± 0.001	0.958 ± 0.001	0.949 ± 0.002	0.936 ± 0.002
ipsweep	0.964 ± 0.004	**0.972** ± 0.003	**0.972** ± 0.003	0.968 ± 0.004	0.966 ± 0.004	0.963 ± 0.004	0.960 ± 0.005	0.946 ± 0.004	0.931 ± 0.004	0.900 ± 0.005
portsweep	0.945 ± 0.005	**0.953** ± 0.004	**0.953** ± 0.004	**0.953** ± 0.004	**0.953** ± 0.004	**0.953** ± 0.004	0.952 ± 0.004	0.944 ± 0.004	0.928 ± 0.004	0.918 ± 0.002
nmap	0.825 ± 0.005	**0.863** ± 0.011	0.833 ± 0.007	0.832 ± 0.006	0.832 ± 0.006	0.832 ± 0.006	0.829 ± 0.007	0.818 ± 0.007	0.792 ± 0.006	0.718 ± 0.020
back	0.616 ± 0.017	**0.954** ± 0.008	0.826 ± 0.018	0.684 ± 0.010	0.648 ± 0.013	0.563 ± 0.011	0.546 ± 0.008	0.529 ± 0.009	0.510 ± 0.010	0.500 ± 0.011
warezclient	0.801 ± 0.084	**0.878** ± 0.009	0.875 ± 0.008	0.874 ± 0.007	0.749 ± 0.092	0.593 ± 0.019	0.512 ± 0.012	0.455 ± 0.017	0.254 ± 0.033	0.005 ± 0.007
teardrop	**0.959** ± 0.009	0.957 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.951 ± 0.008	0.933 ± 0.010	0.914 ± 0.011
normal	0.988 ± 0.001	0.964 ± 0.002	0.982 ± 0.001	0.987 ± 0.001	0.990 ± 0.001	0.993 ± 0.001	0.996 ± 0.001	0.997 ± 0.000	0.998 ± 0.000	**0.999** ± 0.000
overall	0.981 ± 0.001	0.982 ± 0.000	**0.984** ± 0.000	0.983 ± 0.001	0.982 ± 0.001	0.980 ± 0.000	0.980 ± 0.001	0.977 ± 0.000	0.973 ± 0.000	0.967 ± 0.000

ACC	Raw data	**D5**								
		$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$
smurf	0.999 ± 0.000	**1.000** ± 0.001	**1.000** ± 0.001	**1.000** ± 0.001	**1.000** ± 0.001	**1.000** ± 0.001	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000
neptune	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
satan	0.976 ± 0.001	**0.996** ± 0.001	**0.996** ± 0.000	0.994 ± 0.001	0.993 ± 0.001	0.989 ± 0.001	0.978 ± 0.001	0.969 ± 0.001	0.963 ± 0.001	0.951 ± 0.001
ipsweep	0.964 ± 0.004	**0.971** ± 0.003	**0.971** ± 0.003	**0.971** ± 0.003	**0.971** ± 0.003	**0.971** ± 0.003	**0.971** ± 0.003	0.964 ± 0.004	0.961 ± 0.004	0.944 ± 0.005
portsweep	0.945 ± 0.005	**0.953** ± 0.003	**0.953** ± 0.003	**0.953** ± 0.003	**0.953** ± 0.003	**0.953** ± 0.003	0.952 ± 0.004	0.952 ± 0.004	0.952 ± 0.004	0.951 ± 0.003
nmap	0.825 ± 0.005	**0.863** ± 0.006	**0.863** ± 0.006	0.862 ± 0.008	0.839 ± 0.008	0.831 ± 0.006	0.831 ± 0.006	0.831 ± 0.006	0.831 ± 0.006	0.808 ± 0.007
back	0.616 ± 0.017	**0.994** ± 0.002	**0.994** ± 0.002	**0.994** ± 0.002	0.992 ± 0.002	0.987 ± 0.003	0.804 ± 0.018	0.651 ± 0.012	0.546 ± 0.008	0.510 ± 0.010
warezclient	0.801 ± 0.084	**0.903** ± 0.010	**0.903** ± 0.010	0.890 ± 0.011	0.884 ± 0.011	0.879 ± 0.010	0.874 ± 0.010	0.618 ± 0.035	0.506 ± 0.013	0.362 ± 0.020
teardrop	0.959 ± 0.009	**0.963** ± 0.006	**0.963** ± 0.006	0.962 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007	0.956 ± 0.007
normal	0.988 ± 0.001	0.000 ± 0.000	0.014 ± 0.004	0.110 ± 0.014	0.200 ± 0.036	0.422 ± 0.085	0.986 ± 0.001	0.991 ± 0.001	0.996 ± 0.001	**0.998** ± 0.000
overall	0.981 ± 0.001	0.742 ± 0.000	0.746 ± 0.001	0.770 ± 0.004	0.792 ± 0.009	0.846 ± 0.021	**0.984** ± 0.000	0.981 ± 0.001	0.980 ± 0.001	0.976 ± 0.000

ACC	Raw data	**SubD1**							
		$\varepsilon = 0.1$		$\varepsilon = 0.2$		$\varepsilon = 0.3$		$\varepsilon = 0.4$	
		L	R	L	R	L	R	L	R
smurf	**0.999** $\pm$ 0.000	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006	0.022 $\pm$ 0.006
neptune	**1.000** $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
satan	**0.976** $\pm$ 0.001	0.084 $\pm$ 0.006	0.087 $\pm$ 0.006	0.084 $\pm$ 0.006	0.087 $\pm$ 0.006	0.082 $\pm$ 0.006	0.086 $\pm$ 0.006	0.081 $\pm$ 0.006	0.085 $\pm$ 0.006
ipsweep	**0.964** $\pm$ 0.004	0.041 $\pm$ 0.002	0.050 $\pm$ 0.002	0.041 $\pm$ 0.002	0.050 $\pm$ 0.002	0.041 $\pm$ 0.002	0.050 $\pm$ 0.002	0.041 $\pm$ 0.002	0.050 $\pm$ 0.002
portsweep	**0.945** $\pm$ 0.005	0.032 $\pm$ 0.005	0.035 $\pm$ 0.004	0.032 $\pm$ 0.005	0.035 $\pm$ 0.004	0.032 $\pm$ 0.005	0.035 $\pm$ 0.004	0.032 $\pm$ 0.005	0.035 $\pm$ 0.004
nmap	**0.825** $\pm$ 0.005	0.074 $\pm$ 0.026	0.337 $\pm$ 0.055	0.074 $\pm$ 0.026	0.337 $\pm$ 0.055	0.074 $\pm$ 0.026	0.337 $\pm$ 0.055	0.052 $\pm$ 0.023	0.323 $\pm$ 0.056
back	**0.616** $\pm$ 0.017	0.519 $\pm$ 0.017	0.536 $\pm$ 0.017	0.519 $\pm$ 0.017	0.536 $\pm$ 0.017	0.518 $\pm$ 0.018	0.536 $\pm$ 0.017	0.518 $\pm$ 0.018	0.535 $\pm$ 0.017
warezclient	**0.801** $\pm$ 0.084	0.431 $\pm$ 0.018	0.798 $\pm$ 0.044	0.431 $\pm$ 0.018	0.798 $\pm$ 0.044	0.420 $\pm$ 0.021	0.796 $\pm$ 0.044	0.412 $\pm$ 0.021	0.780 $\pm$ 0.047
teardrop	0.959 $\pm$ 0.009	**1.000** $\pm$ 0.000	0.999 $\pm$ 0.001	**1.000** $\pm$ 0.000	0.999 $\pm$ 0.001	0.998 $\pm$ 0.001	0.999 $\pm$ 0.001	0.993 $\pm$ 0.003	0.993 $\pm$ 0.002
normal	**0.988** $\pm$ 0.001	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.014 $\pm$ 0.004	0.009 $\pm$ 0.003	0.111 $\pm$ 0.014	0.097 $\pm$ 0.014	0.197 $\pm$ 0.035	0.187 $\pm$ 0.035
overall	**0.981** $\pm$ 0.001	0.033 $\pm$ 0.001	0.039 $\pm$ 0.002	0.036 $\pm$ 0.002	0.042 $\pm$ 0.002	0.060 $\pm$ 0.003	0.064 $\pm$ 0.004	0.081 $\pm$ 0.009	0.085 $\pm$ 0.008

ACC	Raw data	**SubD2**							
		$\varepsilon = 0.1$		$\varepsilon = 0.2$		$\varepsilon = 0.3$		$\varepsilon = 0.4$	
		L	R	L	R	L	R	L	R
smurf	**0.999** $\pm$ 0.000	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006	0.023 $\pm$ 0.006
neptune	**1.000** $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
satan	**0.976** $\pm$ 0.001	0.097 $\pm$ 0.006	0.098 $\pm$ 0.006	0.096 $\pm$ 0.006	0.097 $\pm$ 0.006	0.092 $\pm$ 0.006	0.093 $\pm$ 0.006	0.092 $\pm$ 0.006	0.092 $\pm$ 0.006
ipsweep	**0.964** $\pm$ 0.004	0.061 $\pm$ 0.003	0.060 $\pm$ 0.003	0.061 $\pm$ 0.003	0.060 $\pm$ 0.003	0.061 $\pm$ 0.003	0.060 $\pm$ 0.003	0.060 $\pm$ 0.003	0.060 $\pm$ 0.003
portsweep	**0.945** $\pm$ 0.005	0.040 $\pm$ 0.003	0.039 $\pm$ 0.003	0.040 $\pm$ 0.003	0.039 $\pm$ 0.003	0.040 $\pm$ 0.003	0.039 $\pm$ 0.003	0.040 $\pm$ 0.003	0.039 $\pm$ 0.003
nmap	**0.825** $\pm$ 0.005	0.386 $\pm$ 0.061	0.370 $\pm$ 0.061	0.381 $\pm$ 0.061	0.370 $\pm$ 0.060	0.381 $\pm$ 0.061	0.370 $\pm$ 0.060	0.381 $\pm$ 0.061	0.370 $\pm$ 0.060
back	**0.616** $\pm$ 0.017	0.536 $\pm$ 0.018	0.536 $\pm$ 0.018	0.534 $\pm$ 0.018	0.535 $\pm$ 0.018	0.529 $\pm$ 0.018	0.531 $\pm$ 0.018	0.523 $\pm$ 0.018	0.526 $\pm$ 0.018
warezclient	0.801 $\pm$ 0.084	0.915 $\pm$ 0.018	**0.920** $\pm$ 0.017	0.897 $\pm$ 0.018	0.906 $\pm$ 0.017	0.868 $\pm$ 0.016	0.874 $\pm$ 0.019	0.868 $\pm$ 0.016	0.869 $\pm$ 0.017
teardrop	0.959 $\pm$ 0.009	0.973 $\pm$ 0.005	**0.976** $\pm$ 0.013	0.973 $\pm$ 0.005	0.975 $\pm$ 0.013	0.973 $\pm$ 0.005	0.974 $\pm$ 0.012	0.973 $\pm$ 0.005	0.974 $\pm$ 0.012
normal	**0.988** $\pm$ 0.001	0.872 $\pm$ 0.005	0.861 $\pm$ 0.004	0.915 $\pm$ 0.006	0.896 $\pm$ 0.005	0.937 $\pm$ 0.001	0.930 $\pm$ 0.001	0.947 $\pm$ 0.004	0.939 $\pm$ 0.002
overall	**0.981** $\pm$ 0.001	0.261 $\pm$ 0.001	0.258 $\pm$ 0.002	0.271 $\pm$ 0.002	0.266 $\pm$ 0.002	0.276 $\pm$ 0.001	0.274 $\pm$ 0.002	0.278 $\pm$ 0.002	0.276 $\pm$ 0.002

ACC	Raw data	**SupD1**							
		$\varepsilon = 0.1$		$\varepsilon = 0.2$		$\varepsilon = 0.3$		$\varepsilon = 0.4$	
		L	R	L	R	L	R	L	R
smurf	0.999 $\pm$ 0.000	**1.000** $\pm$ 0.001	0.999 $\pm$ 0.000	0.999 $\pm$ 0.000	0.999 $\pm$ 0.000	0.999 $\pm$ 0.000	0.999 $\pm$ 0.000	0.999 $\pm$ 0.000	0.999 $\pm$ 0.000
neptune	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000
satan	0.976 $\pm$ 0.001	0.994 $\pm$ 0.001	**0.995** $\pm$ 0.001	0.992 $\pm$ 0.000	0.992 $\pm$ 0.001	0.985 $\pm$ 0.001	0.985 $\pm$ 0.001	0.980 $\pm$ 0.001	0.980 $\pm$ 0.001
ipsweep	0.964 $\pm$ 0.004	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	0.971 $\pm$ 0.003	0.972 $\pm$ 0.003	0.971 $\pm$ 0.003
portsweep	0.945 $\pm$ 0.005	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.003	**0.953** $\pm$ 0.003	**0.953** $\pm$ 0.003	**0.953** $\pm$ 0.003
nmap	0.825 $\pm$ 0.005	0.866 $\pm$ 0.006	**0.876** $\pm$ 0.006	0.865 $\pm$ 0.006	**0.876** $\pm$ 0.006	0.849 $\pm$ 0.009	0.849 $\pm$ 0.009	0.834 $\pm$ 0.009	0.833 $\pm$ 0.008
back	0.616 $\pm$ 0.017	**0.985** $\pm$ 0.004	0.984 $\pm$ 0.004	0.958 $\pm$ 0.007	0.957 $\pm$ 0.007	0.852 $\pm$ 0.013	0.849 $\pm$ 0.013	0.703 $\pm$ 0.016	0.702 $\pm$ 0.016
warezclient	0.801 $\pm$ 0.084	**0.903** $\pm$ 0.013	0.890 $\pm$ 0.014	0.881 $\pm$ 0.009	0.880 $\pm$ 0.009	0.881 $\pm$ 0.009	0.878 $\pm$ 0.009	0.878 $\pm$ 0.009	0.876 $\pm$ 0.008
teardrop	0.959 $\pm$ 0.009	**0.962** $\pm$ 0.006	0.957 $\pm$ 0.007	0.957 $\pm$ 0.007	0.957 $\pm$ 0.007	0.957 $\pm$ 0.007	0.957 $\pm$ 0.007	0.956 $\pm$ 0.007	0.956 $\pm$ 0.007
normal	**0.988** $\pm$ 0.001	0.903 $\pm$ 0.008	0.927 $\pm$ 0.003	0.942 $\pm$ 0.001	0.952 $\pm$ 0.003	0.965 $\pm$ 0.001	0.975 $\pm$ 0.001	0.982 $\pm$ 0.003	0.985 $\pm$ 0.001
overall	0.981 $\pm$ 0.001	0.968 $\pm$ 0.002	0.974 $\pm$ 0.001	0.977 $\pm$ 0.000	0.979 $\pm$ 0.001	0.981 $\pm$ 0.001	**0.983** $\pm$ 0.001	0.982 $\pm$ 0.001	**0.983** $\pm$ 0.001

ACC	Raw data	**SupD2**							
		$\varepsilon = 0.1$		$\varepsilon = 0.2$		$\varepsilon = 0.3$		$\varepsilon = 0.4$	
		L	R	L	R	L	R	L	R
smurf	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000	**0.999** $\pm$ 0.000
neptune	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000
satan	0.976 $\pm$ 0.001	**0.993** $\pm$ 0.001	0.992 $\pm$ 0.000	0.992 $\pm$ 0.001	0.985 $\pm$ 0.001	0.986 $\pm$ 0.001	0.980 $\pm$ 0.001	0.981 $\pm$ 0.001	0.973 $\pm$ 0.001
ipsweep	0.964 $\pm$ 0.004	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	**0.972** $\pm$ 0.003	0.971 $\pm$ 0.004	**0.972** $\pm$ 0.003	0.966 $\pm$ 0.004
portsweep	0.945 $\pm$ 0.005	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.003	**0.953** $\pm$ 0.004	**0.953** $\pm$ 0.003	**0.953** $\pm$ 0.003	0.952 $\pm$ 0.004
nmap	0.825 $\pm$ 0.005	**0.876** $\pm$ 0.006	0.859 $\pm$ 0.008	0.864 $\pm$ 0.008	0.833 $\pm$ 0.007	0.837 $\pm$ 0.009	0.832 $\pm$ 0.006	0.832 $\pm$ 0.006	0.832 $\pm$ 0.006
back	0.616 $\pm$ 0.017	**0.988** $\pm$ 0.002	0.976 $\pm$ 0.004	0.972 $\pm$ 0.005	0.911 $\pm$ 0.016	0.930 $\pm$ 0.013	0.731 $\pm$ 0.013	0.802 $\pm$ 0.017	0.668 $\pm$ 0.011
warezclient	0.801 $\pm$ 0.084	**0.882** $\pm$ 0.010	0.880 $\pm$ 0.009	0.879 $\pm$ 0.009	0.877 $\pm$ 0.008	0.876 $\pm$ 0.008	0.876 $\pm$ 0.008	0.875 $\pm$ 0.008	0.796 $\pm$ 0.089
teardrop	**0.959** $\pm$ 0.009	0.957 $\pm$ 0.007	0.957 $\pm$ 0.007	0.957 $\pm$ 0.007	0.956 $\pm$ 0.007	0.956 $\pm$ 0.007	0.956 $\pm$ 0.007	0.956 $\pm$ 0.007	0.956 $\pm$ 0.007
normal	0.988 $\pm$ 0.001	0.933 $\pm$ 0.001	0.946 $\pm$ 0.002	0.953 $\pm$ 0.004	0.976 $\pm$ 0.001	0.975 $\pm$ 0.001	0.986 $\pm$ 0.001	0.985 $\pm$ 0.001	**0.989** $\pm$ 0.001
overall	0.981 $\pm$ 0.001	0.975 $\pm$ 0.001	0.978 $\pm$ 0.001	0.980 $\pm$ 0.001	**0.984** $\pm$ 0.001	**0.984** $\pm$ 0.000	**0.984** $\pm$ 0.001	**0.984** $\pm$ 0.000	0.982 $\pm$ 0.000