Decomposing Training Data to Improve Network Intrusion Detection Performance

Roberto Saia, Alessandro Sebastian Podda, Gianni Fenu and Riccardo Balia
Department of Mathematics and Computer Science,
University of Cagliari, Via Ospedale 72, 09124 Cagliari, Italy

Keywords: Intrusion Detection, Security, Networking, Data Decomposition, Algorithms.

Abstract: Anyone working in the field of network intrusion detection has been able to observe how it involves an ever-increasing number of techniques and strategies aimed to overcome the issues that affect the state-of-the-art solutions. Data unbalance and heterogeneity are only some representative examples of them, and each misclassification made in this context could have enormous repercussions in different crucial areas such as, for instance, financial, privacy, and public reputation. This happens because the current scenario is characterized by a huge number of public and private network-based services. The idea behind the proposed work is decomposing the canonical classification process into several sub-processes, where the final classification depends on all the sub-processes results, plus the canonical one. The proposed Training Data Decomposition (TDD) strategy is applied on the training datasets, where it applies a decomposition into regions, according to a defined number of events and features. The reason that leads this process is related to the observation that the same network event could be evaluated in a different manner, when it is evaluated in different time periods and/or when it involves different features. According to this observation, the proposed approach adopts different classification models, each of them trained in a different data region characterized by different time periods and features, classifying the event both on the basis of all model results, and on the basis of the canonical strategy that involves all data.

1 INTRODUCTION

The exponential growth of network-based services, now even more impressive due to the COVID-19 (Chang and Meyerhoefer, 2020; Dhawan, 2020; Rapanta et al., 2020) emergency, has made the problem of their security a central aspect of the everyday life, public and private. The high degree of heterogeneity (Zuech et al., 2015) that characterizes network events, in terms both of type and behavior, makes the detection of attacks a very difficult task. This troubling scenario is made even more difficult by considering that many attacks are actually conducted by operating in apparently legitimate ways (i.e., they are characterized by the same patterns of normal network activities (Carta et al., 2020b)), whereas other attacks are being detected for the first time and therefore we have no knowledge of them (zero-days attacks (Radhakrishnan et al., 2019)), and everything is further worsened by the high dynamism of the network scenarios.

For this reason, tools widely used in the past with good results, such as the Intrusion Detection Systems (IDSs) (Tidjon et al., 2019), nowadays appear unable to face the new threats with the same effectiveness. This is a big problem, since there is the need to ensure a high level of security to many crucial services such as, for example, those related to the finance (Wazid et al., 2019), health (Yüksel et al., 2017), and education (Luker and Petersen, 2003).

To cope with the new threats, approaches different from the canonical ones have therefore been sought, integrating increasingly sophisticated technologies and strategies that involve areas such as machine learning (Gao et al., 2019), artificial intelligence (Kanimozhi and Jacob, 2019), neural networks (Le et al., 2019), and so on, also by combining them to improve the intrusion detection performance (Li and Lu, 2019; Meyer and Labit, 2020).

The idea on which the proposed strategy revolves was born according to many literature works in this field (Carta et al., 2020b; Saia et al., 2018b; Saia et al., 2019; Saia et al., 2020), where we trivially observed that the data used to train an evaluation model refers to different period of time, as well as the features that characterize each event refer to different aspect of it.

Saia, R., Podda, A., Fenu, G. and Balia, R.
Decomposing Training Data to Improve Network Intrusion Detection Performance.
DOI: 10.5220/0010661400003064
In *Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2021) - Volume 1: KDIR*, pages 241-248
ISBN: 978-989-758-533-3; ISSN: 2184-3228

Premising that the data to which we refer are those collected by network security devices (e.g., an IDS), on the basis of the previous observation, we thought to divide the classification process into several subprocesses, then on several evaluation models, each of them trained on a different part of the dataset, in terms of events and features. We believe that such a strategy can be able to mitigate the event heterogeneity problem, classifying them on the basis of considerations taken on different parts of the training data, therefore on the basis of different time periods and event characteristics, rather than based on a model trained on the entire dataset.

In more detail, in this work we propose a Training Data Decomposition (TDD) strategy to intrusion detection, which is based on the subdivision of the training dataset in several regions, according to a number of rows and columns that bound it in a certain number of events (rows) and features (columns). Each region is used to train a different evaluation model, and the final classification of a new event is given by ensembling all the models through a majority voting criterion.

The main scientific contribution related to the proposed work can be summarize as follows:

- formalization of a Training Data Decomposition (TDD) strategy aimed to bound the training set on the basis of a certain number of events (rows) and features (columns);
- formalization of the comparison process between an unevaluated network event and a series of evaluation models defined on the basis of the proposed TDD strategy, along with the formalization of the classification criterion;
- formalization of a classification algorithm able to exploit the TDD strategy in order to classify each new network event as *normal* or *intrusion*.

The remainder of this paper has been structured in the following way: Section 2 discusses the background and related works of the intrusion detection domain, discussing also about the most suitable evaluation metrics; Section 3 provides details about the idea behind the proposed strategy, beside its practical implementation; Section 4 ends this work with some remarks on the proposed strategy, making mention of future research directions.

2 BACKGROUND AND RELATED WORK

The intrusion detection term has been introduced for the first time in the eighties (Anderson, 1980), where in his technical report the author tried to define a kind of guideline to improve the computer security auditing and surveillance capability of the computer systems. In the following years the literature proposed numerous works focused on the intrusion detection topic, both of a theoretical nature, where aspects such as their taxonomy have been discussed (Axelsson, 2000), and practical application of them (Khraisat et al., 2019), up to the present days, where the discussion has extended to recent areas, such as that of the Internet of Things (IoT) (Zarpelão et al., 2017), the cloud computing (Modi et al., 2013), and the smart cities (Aloqaily et al., 2019).

The concept of intrusion leads back to a series of attacks based on techniques and/or strategies that evolve over time, an activity whose effectiveness can depend on the skill of a human operator (Latha and Prakash, 2017) or on a specific software/malware (Rehman et al., 2011), or both of them. To this must also be added strategies not directly related to a direct attack, such as, for instance, the *social engineering* (Salahdine and Kaabouch, 2019) one.

An IDS is aimed to detect and identify unauthorized network activities in a network, and it can perform this activity by following different approaches. On the basis of the literature, the most common of them are: *anomaly-based*, *signature-based*, *specification-based*, and *hybrid-based*.

In more detail: the *anomaly-based* analyzes and classifies the network events without comparing them to those in a dataset of known event patterns, since it adopts a heuristic/rules-based strategy, detecting the intrusions on the basis of their atypical network activity (Samrin and Vasumathi, 2017); the *signature-based* works by comparing each detected network event to those in a dataset of known event patterns (Bronte et al., 2016); the *specification-based* operates by inspecting the network protocols, with the aim to identify non canonical sequences of commands that can be part of an attack (Liao et al., 2013); the *hybrid-based* approach can be considered a combination of one or more of the aforementioned approaches (Li et al., 2005).

On the basis of the current literature, we can also summarize the open problems that affect the different intrusion detection approaches: the *Anomaly-based* is able to identify novel form of network attacks (zero-days), as well as the anomalous exploitation of privileges, but it can not be considered an effective approach, due the high dynamism of the network scenario and the high response-time; the *Signature-based* well works in the context of known attacks or their variations, but it is not able to inspect the involved protocols, generating also an high computational load; the *Specification-based* is able to inspect the involved protocols, detecting their anomalous exploitation, but it is not able to distinguish those attacks characterized by the same behavior of a legitimate network activity and, in addition, it presents an high computational cost related to the protocols inspection and tracing; the *Hybrid-based* presents the same pros and cons of the combined approaches.

According to the approach and placement in the network area, an IDS can be also classified in four (most common) categories: *Host-based*, *Network-based*, *Network-node-based*, and *Distributed-based*.

In more detail: the *Host-based Intrusion Detection Systems* (HIDSs) (Jose et al., 2018) adopts several hosts to detect the network activity, detecting the attacks, by comparing the events to a series of known patterns (signature-based approach); the *Network-based Intrusion Detection Systems* (NIDSs) (Mazini et al., 2019) adopts a single host to detect the network activity, it exploits a database of known patterns, analyzing only the events characterized by unknown patterns (hybrid signature-based and analysis-based approach); the *Network-Node-based Intrusion Detection Systems* (NNIDSs) (Potluri and Diedrich, 2016) adopts a single host strategically placed in the network, operating by combining the HIDS and NIDS strategies; the *Distributed-based Intrusion Detection Systems* (DIDSs) (Amrita, 2018) adopts an hybrid strategy based on the combination of all the aforementioned ones.

Moreover, recently, some security features offered by techniques typically exploited in other areas have been started to be taken into consideration for facing network security tasks, such as, for instance, those leveraging on the blockchain primitives (Vieira et al., 2020; Longo et al., 2020).

2.1 Performance Evaluation

A preliminary consideration regarding the evaluation metrics used in this domain is related to the fact that, similar to some other domains (Saia et al., 2017; Carta et al., 2020a; Saia and Carta, 2017a; Saia, 2017; Saia and Carta, 2016; Saia et al., 2018a; Saia and Carta, 2017b), the involved data are usually characterized by a high degree of imbalance (in the intrusion detection context the minority class is the *intrusion* one), requiring assessment metrics that are not biased by this characteristic.

Since, in the literature, an intrusion detection task is commonly expressed in terms of a binary problem, then addressed as a classification task (Chuang et al., 2019), the best approach requires to adopt multiple evaluation metrics, in order to get a reliable evaluation of the effectiveness of a classification model. For this reason, simple metrics based on the *confusion-matrix*, e.g., *accuracy*, *sensitivity*, and *specificity*, and more sophisticated ones, such as those based on the *Receiver Operating Characteristic* (ROC) curve, e.g., the *Area Under the Receiver Operating Characteristic curve* (AUC), are often combined in the literature (Munaiah et al., 2016).

Moreover, considering that the main objective of an intrusion detection system is the correct identification and classification of the negative cases (*intrusions*), as their misclassification would have a higher cost than that of the positive ones (*normals*), many of the works in the literature use the *specificity* and the AUC metrics.

3 PROPOSED STRATEGY

This section provides the formal notation adopted in this work, along with the definition of the problem to face, and the implementation of the proposed strategy.

3.1 Preliminary Notation

Premising that we adopted the notation $|set|$ to indicate the cardinality of a *set*, we denote as $E = \{e_1, e_2, \ldots, e_N\}$ a series of network events composed by:

- a subset $E^+ = \{e_1^+, e_2^+, \ldots, e_X^+\}$ of *normal* events, then $E^+ \subseteq E$;
- a subset $E^- = \{e_1^-, e_2^-, \ldots, e_Y^-\}$ of *intrusion* events, then $E^- \subseteq E$;
- a subset $\hat{E} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_M\}$ of unclassified events, then $\hat{E} \subseteq E$.

So we have that $E = (E^+ \cup E^- \cup \hat{E})$, and each event $e \in E$ is characterized by the features in the set $F = \{f_1, f_2, \ldots, f_W\}$, and it can belong to one of the classes in the set $C = \{normal, intrusion\}$. We also formalize:

- the *training set* $T = \{e_1, e_2, \ldots, e_K\}$ given by $E^+ \cup E^-$;
- the possibility to divide T into $R = \{r_1, r_2, \ldots, r_Z\}$ regions, according to the T events (set rows) and features (set columns);
- the regions definition operation as $R_{(ER,FC)}$, with ER the number of *Event Rows*, and FC the number of *Feature Columns*, then $|R| = Z = (ER \times FC)$.

From this it follows that:

- generalizing on set E, each region can be composed by $\frac{N}{ER}$ events and $\frac{W}{FC}$ features, since $|E| = N$ and $|F| = W$;
- the bounds of ER and FC are, respectively, $1 \leq ER \leq |T|$ and $1 \leq FC \leq |F|$, but it must be considered that $ER = FC = 1$ defines the canonical data configuration, where all the events and the features are involved, and that each region defined according the ER value must contain samples (events) of both classes in C, in order to allow us to perform the training process of an evaluation model.

3.2 Problem Statement

We face the intrusion detection in terms of binary classification, according to the classes in the set C (i.e., *normal* and *intrusion*). Hence, the problem can be formalized as shown in Equation 1, where Ψ denotes a generic intrusion detection approach, whereas $eval(\hat{e}, \Psi)$ is the evaluation function of the event $\hat{e}$, which returns 1 when a classification has been performed correctly, 0 otherwise. This allows us to express this problem in terms of maximization of the ω value, since it is given by the sum of the correct classifications (then the ω upper bound is $|\hat{E}|$).

$$\max_{0 \leq \omega \leq |\hat{E}|} \omega = \sum_{m=1}^{|\hat{E}|} eval(\hat{e}_m, \Psi) \tag{1}$$

3.3 Strategy Overview

We now briefly introduce the TDD strategy, a domain-specific case of the *bootstrap aggregation* (Breiman, 1996) strategy, here proposed as an alternative to a canonical intrusion detection evaluation model that exploits, during the training process, all the data that have been collected and correctly classified in the past, involving all the events and features.

Indeed, we propose to combine different evaluation models in a *fusion fashion*, each of them trained on a different region of the training set. The regions are bounded in terms of events and features, with the aim to selectively capture the properties of each sub-region by focusing on specific periods of *time* (rows of events) and *behavior* (columns of features). This has been made according to the simple scheme reported in the following, where four regions of the training set T, with two events and two features each one (one of them has been highlighted), have been selected by way of example:

$$\mathbf{T} = \begin{array}{|cccc c|} \hline \multicolumn{4}{c}{\longleftarrow \textit{ behavior } \longrightarrow} & \\ e_1(f_1) & e_1(f_2) & e_1(f_{\dots}) & e_1(f_W) & \uparrow \\ e_2(f_1) & e_2(f_2) & e_2(f_{\dots}) & e_2(f_W) & \textit{time} \\ \vdots & \vdots & \ddots & \vdots & \downarrow \\ e_N(f_1) & e_N(f_2) & e_N(f_{\dots}) & e_N(f_W) & \\ \hline \end{array}$$

This leads to the division of the training process into several sub-processes, each of them based on a different part of the dataset, in terms of events and features. In other words, we speculate that such a strategy is able to mitigate the issues related to the event heterogeneity, since the classification of the new events is now based on decisions taken on the basis of different time periods and characteristics (aggregated according to a criterion), rather than on the entire dataset.

3.4 Strategy Formalization

The problem formalized in Equation 1 needs to be arranged according to the subdivision into regions we introduced, substantially by subdividing the evaluation process into Z sub-processes (i.e., $|R| = Z$). More formally, a generic intrusion detection approach Ψ is used Z times, and the final event classification will be determined on the basis of all the classifications performed by these approaches, as shown in the example of Equation 2, where we hypothesized $K = 4$, $W = 4$, $ER = 2$, and $FC = 2$, thus subdividing the training set T into $|R| = Z = (2 \times 2) = 4$ regions, each of them composed by $\frac{K}{ER} = \frac{4}{2} = 2$ events and $\frac{W}{FC} = \frac{4}{2} = 2$ features. This generates the four m_1, m_2, m_3, m_4 evaluation models.

$$R_{(2,2)} = \left[\begin{array}{c|c} r_1 & r_2 \\ \hline r_3 & r_4 \end{array}\right] = \left[\begin{array}{cc|cc} f_{1,1} & f_{2,1} & f_{3,1} & f_{4,1} \\ f_{1,2} & f_{2,2} & f_{3,2} & f_{4,2} \\ \hline f_{1,3} & f_{2,3} & f_{3,3} & f_{4,3} \\ f_{1,4} & f_{2,4} & f_{3,4} & f_{4,4} \end{array}\right] \Rightarrow \left[\begin{array}{c|c} m_1 & m_2 \\ \hline m_3 & m_4 \end{array}\right] \tag{2}$$

Otherwise speaking, the process of training of the evaluation model m of a classification algorithm is performed by using the events and features in each r_1, r_2, r_3, r_4 regions of the Equation 2, generating the four m_1, m_2, m_3, m_4 evaluation models.

Assuming the evaluation of a new event $\hat{e} \in \hat{E}$, which on the basis of the scenario we hypothesized will be composed by the f_1, f_2, f_3, f_4 features, its evaluation and classification will be performed by comparing (we denoted this operation as $\Leftrightarrow$) it to all evaluation models trained on the different regions, independently, and this process returns four classification

c_1, c_2, c_3, c_4, according to the criterion shown in Equation 3.

$$c_1 = \begin{bmatrix} m_1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} f_1 \; f_2 \end{bmatrix} \;\; c_2 = \begin{bmatrix} m_2 \end{bmatrix} \Leftrightarrow \begin{bmatrix} f_3 \; f_4 \end{bmatrix}$$
$$c_3 = \begin{bmatrix} m_3 \end{bmatrix} \Leftrightarrow \begin{bmatrix} f_1 \; f_2 \end{bmatrix} \;\; c_4 = \begin{bmatrix} m_4 \end{bmatrix} \Leftrightarrow \begin{bmatrix} f_3 \; f_4 \end{bmatrix} \quad (3)$$

3.4.1 Padding Rule

Considering that the number of regions given by the values of FC and ER may not exactly partition the sets F and T in the cases reported in Equation 4, we need to formalize a *padding rule* aimed to face this problem.

$$\begin{array}{c} (|F| \bmod FC) \neq 0 \\ (|T| \bmod ER) \neq 0 \end{array} \quad (4)$$

Introducing the notation $\mu_1 = (|F| \bmod FC)$ and $\mu_2 = (|T| \bmod ER)$, according to the preliminary notation provided in Section 3.1 (i.e., $F = \{f_1, f_2, \ldots, f_W\}$ and $T = \{e_1, e_2, \ldots, e_K\}$), we formalize in Equation 5 the *padding rule* pad.

$$\begin{array}{l} \texttt{pad}(F) = \{f_1, f_2, \ldots, f_W, f_{W+1}, f_{W+2}, \ldots, f_{W+\mu_1}\} \\ \textit{with } f_{W+1} = f_{W+2} = \ldots = f_{W+\mu_1} = f_W \\ \\ \texttt{pad}(T) = \{e_1, e_2, \ldots, e_K, e_{K+1}, e_{K+2}, \ldots, e_{K+\mu_2}\} \\ \textit{with } e_{K+1} = e_K, e_{K+2} = e_{K-1}, \ldots = e_{K+\mu_2} = e_{K-\mu_2} \end{array} \quad (5)$$

Practically, with the aim of not significantly altering the involved information during the padding process, the problem has been faced by following two different strategies: (*i*) in the case of F, we operate by duplicating the last column (i.e., the last feature of the set E) μ_1 times; (*ii*) in the case of T we duplicated the last μ_2 rows (i.e., the last events of the set T), reducing the risk that the added events belong to the same class in C. The proposed naive solution does not significantly affect the machine learning process, as it is applied to both training and test data. For simplification reasons, we will always consider this rule applied during the process of definition of the regions (as internal preprocessing step), without referring to it from time to time.

3.4.2 Classification Rule

As previously formalized, excepting for the case $ER = FC = 1$ that bounds a single region (i.e., the canonical data configuration), the process of classification of an event $\hat{e} \in \hat{E}$ involves a series of evaluation models $m_1, m_2, \ldots, m_Z$, whose cardinality depends on the number of regions, since $|R| = Z$. Considering that such models generate $c_1, c_2, \ldots, c_Z$ classifications, this configures one of the two cases reported in Equation 6.

$$\begin{array}{l} \texttt{Case 1}: Z = 2n,\; n \in \mathbb{N} \\ \texttt{Case 2}: Z = 2n-1,\; n \in \mathbb{N} \end{array} \quad (6)$$

Whereas in the `Case 2` an univocal classification of the event is possible on the basis of the majority classification, in the `Case 1` we need to introduce a discriminating element. For this purpose we use a further classification c_{Z+1} obtained by using an evaluation model trained on the entire set E, then we will have the $c_1, c_2, \ldots, c_Z, c_{Z+1}$ classifications.

In more detail, assuming an example scenario related `Case 1`, for instance by using $ER = FC = 2$, which generate the classification models m_1, m_2, m_3, m_4, providing the c_1, c_2, c_3, c_4 classifications (each of them that belongs to a class in the set C, then *normal* or *intrusion*) for an event $\hat{e} \in \hat{E}$, the final classification of the event $\hat{e}$ is given by adding the classification c_5 obtained by training an evaluation model on the entire set T.

A majority criterion is then applied by following the *classification rule* ρ formalized in Equation 7, where c_1 and c_2 denote, respectively, the elements *normal* and *intrusion* of the C set.

$$\rho(\hat{e}) = \begin{cases} c_1, \textit{ if } \sum\limits_{i=1}^{Z} \phi(c_i, c_1) > \sum\limits_{i=1}^{Z} \phi(c_i, c_2) \\ c_2, \textit{ if } \sum\limits_{i=1}^{Z} \phi(c_i, c_1) < \sum\limits_{i=1}^{Z} \phi(c_i, c_2) \\ c_1, \textit{ if } \sum\limits_{i=1}^{Z} \phi(c_i, c_1) = \sum\limits_{i=1}^{Z} \phi(c_i, c_2) \wedge c_{Z+1} = c_1 \\ c_2, \textit{ if } \sum\limits_{i=1}^{Z} \phi(c_i, c_1) = \sum\limits_{i=1}^{Z} \phi(c_i, c_2) \wedge c_{Z+1} = c_2 \end{cases} \quad (7)$$

with

$$\phi(a, b) = \begin{cases} 0, \textit{ if } a \neq b \\ 1, \textit{ if } a = b \end{cases}$$

3.5 Algorithm Definition

On the basis of the proposed strategy, we formalized the Algorithm 1, which is aimed to classify each new network event. It takes as input a classification algorithm α, the set of classified events T (i.e., the training set), the set $\hat{E}$ of events to classify, and the number of rows (ER) and columns (FC) for the regions definition, returning the classification of all events in the set $\hat{E}$.

At Steps from *2* to *4* an evaluation model is trained by using the entire set T, if the numbers of regions (i.e., $Z = |ER \times FC|$) is even. At Step *5* the training test T is processed in order to define the regions, according to the ER and FC values. For each region, at Steps from *6* to *9*, an evaluation model of the algorithm α is trained. The classification of the events in $\hat{E}$ is performed from Step *10* to *21*, where: at Step *11* the features of the event $\hat{e}$ to evaluate are divided into regions, according to the ER and FC values; at Steps from *12* to *15* each region is classified according to the M models previously trained, and an additional classification based on the model trained at Step *3* is added to the set C when the number of regions is even (Steps from *16* to *19*). At Step *20* the event $\hat{e}$ is classified, and the classification is stored in the set κ. The set κ with the classification of all events in the set $\hat{E}$ is finally returned by the algorithm at Step *22*.

Algorithm 1: Classifier algorithm.

```
Require: $\alpha$=Classification algorithm, $T$=Evaluated events, $\hat{E}$=Unevaluated
    events, $ER$=Number of event rows, $FC$=Number of feature columns
Ensure: $\kappa$=Classification of the $\hat{E}$ events
 1: procedure CLASSIFIER($\alpha$, $T$, $\hat{E}$, $ER$, $FC$)
 2:     if $Z$ is even then                      ▷ Verifies if the number of regions is even
 3:         $m'' \leftarrow getTraining(\alpha, T)$ ▷ Trains evaluation model by using the
    whole set $T$
 4:     end if
 5:     $R \leftarrow getRegions(T, ER, FC)$          ▷ Divides training set into regions
 6:     for each $r \in R$ do               ▷ Trains an evaluation model for each regions
 7:         $m \leftarrow getTraining(\alpha, r)$               ▷ Trains evaluation model
 8:         $M.add(m)$                                  ▷ Stores evaluation model
 9:     end for
10:     for each $\hat{e} \in \hat{E}$ do                        ▷ Processes events in $\hat{E}$
11:         $R'' \leftarrow getRegions(\hat{e}, ER, FC)$      ▷ Divides event into regions
12:         for each $m \in M$ do                   ▷ Gets all event classifications
13:             $c \leftarrow getEventClass(m, R'')$  ▷ Classifies event according to
    regions
14:             $C.add(c)$                                  ▷ Stores classification
15:         end for
16:         if $Z$ is even then              ▷ Verifies if the number of regions is even
17:             $c'' \leftarrow getEventClass(m'', \hat{e})$ ▷ Classifies event according to
    the whole set $E$
18:             $C.add(c'')$                       ▷ Add classification to the set $C$
19:         end if
20:         $\kappa.add(getFinalClassification(\hat{e}, C))$     ▷ Gets and store final
    event classification
21:     end for
22:     return $\kappa$                         ▷ Returns classification of $\hat{E}$ events
23: end procedure
```

4 CONCLUSIONS AND FUTURE DIRECTIONS

This work proposes a Training Data Decomposition (TDD) strategy aimed to improve the performance of an intrusion detection system. It is based on the consideration that by dividing the training dataset into several regions, it is possible to characterize different scenarios in terms of time (number of events) and characteristics (features), allowing the definition of a more effective analysis model. Each region is used in order to train an independent evaluation model, and the final classification of each new event is performed by taking into account the classifications of all the involved models, in a fusion fashion regulated by a majority voting criterion. In order to be able to apply the TDD strategy, regardless the used dataset (i.e., different number of events and features), some rules have been also formalized.

As next step, we will experiment the proposed strategy in the context of a real-world dataset such as, for instance, the largely used in the literature NSL-KDD[1] one, which includes a large number of different network events in terms of protocols and intrusions, allowing us to verify the hypothesis behind this work.

REFERENCES

Aloqaily, M., Otoum, S., Al Ridhawi, I., and Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90:101842.

Amrita, K. K. R. (2018). A hybrid intrusion detection system: Integrating hybrid feature selection approach with heterogeneous ensemble of intelligent classifiers. *International Journal of Network Security (IJNS'18)*, 20(1):41–55.

Anderson, J. P. (1980). Computer security threat monitoring and surveillance.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy. Technical report, Technical report.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Bronte, R., Shahriar, H., and Haddad, H. M. (2016). A signature-based intrusion detection system for web applications based on genetic algorithm. In *Proceedings of the 9th International Conference on Security of Information and Networks*, pages 32–39.

Carta, S., Corriga, A., Ferreira, A., Podda, A. S., and Recupero, D. R. (2020a). A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Applied Intelligence*, pages 1–17.

Carta, S., Podda, A. S., Reforgiato Recupero, D. R., and Saia, R. (2020b). A local feature engineering strategy to improve network anomaly detection. *Future Internet*, 12(10):177.

Chang, H.-H. and Meyerhoefer, C. D. (2020). Covid-19 and the demand for online food shopping services: Empirical evidence from taiwan. *American Journal of Agricultural Economics*.

Chuang, H.-M., Huang, H.-Y., Liu, F., and Tsai, C.-H. (2019). Classification of intrusion detection system based on machine learning. In *International Cognitive Cities Conference*, pages 492–498. Springer.

Dhawan, S. (2020). Online learning: A panacea in the time of covid-19 crisis. *Journal of Educational Technology Systems*, 49(1):5–22.

Gao, X., Shan, C., Hu, C., Niu, Z., and Liu, Z. (2019). An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, 7:82512–82521.

[1]https://github.com/defcom17/NSL_KDD

Jose, S., Malathi, D., Reddy, B., and Jayaseeli, D. (2018). A survey on anomaly based host intrusion detection system. In *Journal of Physics: Conference Series*, volume 1000, page 012049. IOP Publishing.

Kanimozhi, V. and Jacob, T. P. (2019). Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0033–0036. IEEE.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):20.

Latha, S. and Prakash, S. J. (2017). A survey on network attacks and intrusion detection systems. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–7. IEEE.

Le, T.-T.-H., Kim, Y., Kim, H., et al. (2019). Network intrusion detection based on novel feature selection model and various recurrent neural networks. *Applied Sciences*, 9(7):1392.

Li, Y. and Lu, Y. (2019). Lstm-ba: Ddos detection approach combining lstm and bayes. In *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, pages 180–185. IEEE.

Li, Z., Das, A., and Zhou, J. (2005). Usaid: Unifying signature-based and anomaly-based intrusion detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 702–712. Springer.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Longo, R., Podda, A. S., and Saia, R. (2020). Analysis of a consensus protocol for extending consistent subchains on the bitcoin blockchain. *Computation*, 8(3):67.

Luker, M. A. and Petersen, R. J. (2003). *Computer and network security in higher education*. Jossey-Bass San Francisco, CA.

Mazini, M., Shirazi, B., and Mahdavi, I. (2019). Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and adaboost algorithms. *Journal of King Saud University-Computer and Information Sciences*, 31(4):541–553.

Meyer, M. L. B. and Labit, Y. (2020). Combining machine learning and behavior analysis techniques for network security. In *2020 International Conference on Information Networking (ICOIN)*, pages 580–583. IEEE.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., and Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of network and computer applications*, 36(1):42–57.

Munaiah, N., Meneely, A., Wilson, R., and Short, B. (2016). Are intrusion detection studies evaluated consistently? a systematic literature review.

Potluri, S. and Diedrich, C. (2016). High performance intrusion detection and prevention systems: A survey. In *ECCWS2016-Proceedings fo the 15th European Conference on Cyber Warfare and Security*, page 260. Academic Conferences and publishing limited.

Radhakrishnan, K., Menon, R. R., and Nath, H. V. (2019). A survey of zero-day malware attacks and its detection methodology. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pages 533–539. IEEE.

Rapanta, C., Botturi, L., Goodyear, P., Guàrdia, L., and Koole, M. (2020). Online university teaching during and after the covid-19 crisis: Refocusing teacher presence and learning activity. *Postdigital Science and Education*, 2(3):923–945.

Rehman, R., Hazarika, G., and Chetia, G. (2011). Malware threats and mitigation strategies: a survey. *Journal of Theoretical and Applied Information Technology*, 29(2):69–73.

Saia, R. (2017). A discrete wavelet transform approach to fraud detection. In *International Conference on Network and System Security*, pages 464–474. Springer.

Saia, R. and Carta, S. (2016). A linear-dependence-based approach to design proactive credit scoring models. In *KDIR*, pages 111–120.

Saia, R. and Carta, S. (2017a). Evaluating credit card transactions in the frequency domain for a proactive fraud detection approach. In *SECRYPT*, pages 335–342.

Saia, R. and Carta, S. (2017b). A fourier spectral pattern analysis to design credit scoring models. In *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, page 18. ACM.

Saia, R., Carta, S., et al. (2017). A frequency-domain-based pattern mining for credit card fraud detection. In *IoTBDS*, pages 386–391.

Saia, R., Carta, S., and Fenu, G. (2018a). A wavelet-based data analysis to credit scoring. In *Proceedings of the 2nd International Conference on Digital Signal Processing*, pages 176–180. ACM.

Saia, R., Carta, S., and Recupero, D. R. (2018b). A probabilistic-driven ensemble approach to perform event classification in intrusion detection system. In *KDIR*, pages 139–146.

Saia, R., Carta, S., Recupero, D. R., and Fenu, G. (2020). A feature space transformation to intrusion detection systems. In *KDIR*. ScitePress.

Saia, R., Carta, S., Recupero, D. R., Fenu, G., and Stanciu, M. (2019). A discretized extended feature space (defs) model to improve the anomaly detection performance in network intrusion detection systems. In *KDIR*, pages 322–329.

Salahdine, F. and Kaabouch, N. (2019). Social engineering attacks: A survey. *Future Internet*, 11(4):89.

Samrin, R. and Vasumathi, D. (2017). Review on anomaly based network intrusion detection system. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 141–147. IEEE.

Tidjon, L. N., Frappier, M., and Mammar, A. (2019). Intrusion detection systems: A cross-domain overview. *IEEE Communications Surveys & Tutorials*, 21(4):3639–3681.

Vieira, E., Ferreira, J., and Bartolomeu, P. C. (2020). Blockchain technologies for iot applications: Use-cases and limitations. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1560–1567. IEEE.

Wazid, M., Zeadally, S., and Das, A. K. (2019). Mobile banking: evolution and threats: malware threats and security solutions. *IEEE Consumer Electronics Magazine*, 8(2):56–60.

Yüksel, B., Küpçü, A., and Özkasap, Ö. (2017). Research issues for privacy and security of electronic health services. *Future Generation Computer Systems*, 68:1–13.

Zarpelão, B. B., Miani, R. S., Kawakani, C. T., and de Alvarenga, S. C. (2017). A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37.

Zuech, R., Khoshgoftaar, T. M., and Wald, R. (2015). Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2(1):3.