

# Identifying Relevant Features of CSE-CIC-IDS2018 Dataset for the Development of an Intrusion Detection System

László Göcs <sup>a,\*</sup>, Zsolt Csaba Johanyák <sup>b</sup>

<sup>a</sup> Department of Information Technology, John von Neumann University, GAMF Faculty of Engineering and Computer Science, Izsáki út 10, Kecskemét

E-mail: [gocs.laszlo@nje.hu](mailto:gocs.laszlo@nje.hu)

<sup>b</sup> Department of Information Technology, John von Neumann University, GAMF Faculty of Engineering and Computer Science, Izsáki út 10, Kecskemét

E-mail: [johanyak.csaba@nje.hu](mailto:johanyak.csaba@nje.hu)

**Abstract.** Intrusion detection systems (IDSs) are essential elements of IT systems. Their key component is a classification module that continuously evaluates some features of the network traffic and identifies possible threats. Its efficiency is greatly affected by the right selection of the features to be monitored. Therefore, the identification of a minimal set of features that are necessary to safely distinguish malicious traffic from benign traffic is indispensable in the course of the development of an IDS. This paper presents the preprocessing and feature selection workflow as well as its results in the case of the CSE-CIC-IDS2018 on AWS dataset, focusing on five attack types. To identify the relevant features, six feature selection methods were applied, and the final ranking of the features was elaborated based on their average score. Next, several subsets of the features were formed based on different ranking threshold values, and each subset was tried with five classification algorithms to determine the optimal feature set for each attack type. During the evaluation, four widely used metrics were taken into consideration.

Keywords: dataset preprocessing, dimension reduction, feature selection, classification, Python, CE-CIC-IDS2018

## 1 Introduction

Nowadays, only automated tools called Intrusion Detection Systems (IDSs) are capable of efficiently detecting attacks against IT systems. They continuously monitor and evaluate the parameters of network packages. An IDS is a software or hardware solution that can detect out-of-the-ordinary packages and activities capable of damaging the computer or even the network. An IDS device monitors traffic passing through network interfaces. As soon as it detects malicious activity, it sends an alarm message to a pre-configured monitoring system that can prevent further attacks by re-configuring network devices such as security appliances or traffic controllers. The IDS is often deployed at the boundary of the trusted network, sometimes even outside the firewall.

---

\* Corresponding author. E-mail: [gocs.laszlo@nje.hu](mailto:gocs.laszlo@nje.hu).

Intrusion detection systems can be categorized in several ways, e.g., by the intrusion detection approach used (anomaly-based or signature-based), by the type of system protected (host, network, hybrid), by the IDS architecture (centralized, distributed), by the source of data used for analysis (network packages, system analysis), by the level of service provided after attack detection (active, passive), and by the timing of analysis (continuous, time interval) [1].

The results being reported in this paper were obtained in the course of an investigation focusing on anomaly-based intrusion detection systems (also called Behavior-based IDSs – BIDSs). These systems operate in two modes (learning and detection). In the learning mode, the system is fed with sensor data that contain typical (normal) network and malicious (attack) data. The classification unit is trained and tested based on the labels associated with the data records. In detection mode, the fully trained classification module aims to determine whether the current activity is harmful to the system or not. The anomaly-based approach has the advantage of being able to adapt quickly and dynamically to unknown attack types. BIDSs can be classified into three main categories based on the way they process data, namely statistical-based, knowledge-based, and computational intelligence-based [2].

The classification module is the most important component of an IDS. Its efficiency and speed are affected to a great extent by the right selection of the features being monitored and used for the classification. The main focus of the research reported in this paper was on determining these features in the case of several network datasets containing normal data as well as data related to different attack types. For each attack type, the rank of each feature was determined based on the average score obtained from the results of the application of several feature selection methods. Next, different classification methods were used to evaluate a series of rank threshold values to determine the optimal threshold and feature set for each attack type.

The rest of the paper is organized as follows. Section 2 contains a literature review. Section 3 presents the used datasets and the preprocessing steps. The applied feature selection methods are described in Section 4. Section 5 describes the classification algorithms used in the course of the evaluation while the results are discussed in Section 6. The conclusions are drawn in Section 7.

## 2 Related works

The sample data based development possibilities of IDS classification modules have been intensively investigated in the last decade. Kurniabudi et al. [3] used Information Gain (IG) method to rank and cluster features of the CICIDS-2017 dataset and then applied Random Forest (RF), Bayes Net (BN), Random Tree (RT), Naive Bayes (NB) and J48 classification algorithms to select the features, which yielded good classification results.

Rahman et al. [4] performed AWID dataset analysis using Support Vector Machine (SVM) and C4.5 as feature selection methods using artificial neural networks (ANN) based classification achieving 99.95% accuracy. Javadpour et al. [5] used Pearson Linear Correlation and IG to select the features of the KDD99 dataset and CART, ANN, Decision Tree, and Random Forest (RF) algorithms for classification. They obtained the best results (99.98% accuracy) using the neural network method. Taher et al. (2019) used correlation and Chi-square-based techniques as feature selection methods for the NSL-KDD dataset, followed by ANN and SVM classification algorithms achieving 94.02% recognition rate [6].

Kocher et al. used the Chi-square approach for dimensionality reduction on the UNSW-NB15 dataset followed by k-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Random Forest, Logistic Regression (LR) and Naive Bayes (NB) algorithms for classification, and resulting in a classifier accuracy of 99.64% [7]. Alkasassbeh [8] used BayesNet, MLP, and SVM machine learning methods, and IG,

ReliefF (RF), and Genetic Search (GS) for feature selection. The best accuracy (99.9%) was achieved with BayesNet and GS.

Thaseen et al. used the Chi-square approach to select features of the NSL KDD dataset and did the classification with an SVM classifier. The proposed model resulted in a high detection rate and a low false alarm rate [9]. Awotunde et al. compared NSL-KDD and UNSW-NB15 datasets using hybrid rule-based feature selection and the DFFNN deep learning algorithm, with a recognition rate of 99.0% for NSL-KDD and 98.9% for UNSW-NB15 [10].

Sasan et al. applied the J48 and Classification & regression Trees (CART) methods for the NSL-KDD dataset using 29 features and achieved 88.23% accuracy [11]. However, the article does not describe how the 29 features were selected. Biswas presented a comparison of feature selection methods (CFS, IGR, PCA) and classifier algorithms (NB, SVM, DT, NN, k-NN) on the NSL-KDD dataset, which shows that the k-NN classifier performs better than the others and among the feature selection methods, IGR feature selection method is better [12].

Shaukat et al. investigated the CICIDS-2017 dataset using CFS and Naive Bayes feature selection methods with MLP and IBK algorithms, which showed that IBK is more accurate than MLP [13]. Malhotra et al. used Naive Bayes, Bayes Net, Logistic, Random Tree, Random Forest, J48, Bagging, OneR, PART, and ZeroR classifiers for the analysis of the NSL-KDD dataset, out of which Random Forest, Bagging, PART, and J48 were the best four in terms of model construction time. However, Random Tree achieved good accuracy in a short time without using feature selection and dimension reduction methods [14].

Krishnaveni et al. used IG, Chi-square, Gain Ratio, Symmetric Uncertainty, and Relief methods for feature selection for Real-Time HoneyPot, NSL-KDD, and Kyoto datasets, and also used SVM, Naive Bayes, Logistic Regression, and Decision Tree classification algorithms [15]. Kumar et al. used CFS, IGF, and GR methods for the feature selection in the case of the NSL-KDD dataset and applied Naive Bayes, J48, and RepTree algorithms for classification. The feature subset identified by GR and Ranker improved the proposed Naive Bayes classification [16].

Pattawaro and Polprasert utilized a feature selection method based on attribute ratio (AR) for the NSL-KDD dataset combined with k-Means clustering and XGBoost classification. The proposed model achieved an accuracy of 84.41% [17]. Tohari et al. worked with k-Nearest Neighbor (k-NN), SVM, and Naive Bayes classifiers on the KDD Cup99, Kyoto 2006, and UNSW-NB15 datasets [18], where the best performance is achieved by SVM with 99.9291% accuracy and 0% false positive rate [18].

### 3 Preprocessing the datasets

The classification module is the core module of an IDS. Usually, it is developed using one or more sample datasets and applying statistical or machine learning techniques. These datasets contain an immense amount of data describing normal (benign) and malicious traffic. The raw data obtained from the sensors usually have to undergo several preprocessing steps until it can be used for the training of the classification module. These steps can be divided into three main phases: data cleaning, data transformation, and data reduction. In the following subsections, after a short introduction of the available and used datasets the preprocessing applied in course of this investigation is presented in detail.

### 3.1 Datasets

The first sample dataset used for IDS training purposes was the famous KDD'99 [19], which has served later as a starting point for the development of several IDS solutions. It contains information about simulated traffic corresponding to normal activities and several attack types (DOS, guesspassword, buffer overflow, remote FTP, synflood, Nmap, rootkit). Subsequently, some other datasets have been also created containing samples of new attack types as well as some additional features. The most relevant datasets are presented in Table 1. The second column presents the attack types covered by the given dataset.

Table 1  
IDS Datasets

Dataset	Attack types included
CSE-CIC-IDS2018 on AWSnet [20]	Bruteforce attack, DoS attack, Web attack, Infiltration attack, Botnet attack, DDoS+PortScan
CIC-IDS2017 [21] [22] [23]	botnet (Ares), cross-site-scripting, DoS (executed through Hulk, GoldenEye, Slowloris, and Slowhttptest), DDoS (executed through LOIC), heartbleed, infiltration, SSH brute force, SQL injection
CIDDS-001 [24] [25]	DoS, port scans (ping-scan, SYN-Scan), SSH brute force
CIDDS-002 [24] [25]	port scans (ACK-Scan, FIN-Scan, ping-Scan, UDP-Scan, SYN-Scan)
UNSW-NB15 [26] [27]	backdoors, DoS, exploits, fuzzers, generic, port scans, reconnaissance, shellcode, spam, worms
UGR'16 [28]	botnet (Neris), DoS, port scans, SSH brute force, spam
TUIDS [29]	botnet (IRC), DDoS (Fraggle flood, Ping flood, RST flood, smurf ICMP flood, SYN flood, UDP flood), port scans (e.g. FIN-Scan, NULL-Scan, UDP-Scan, XMAS-Scan), coordinated port scan, SSH brute force

The dataset used in course of the research reported in this paper is the CSE-CIC-IDS2018 on AWS [20] that was created by the Canadian Institute for Cybersecurity lab. This dataset was chosen because it is one of the most recent ones and meets all the criteria (e.g. total traffic, many attacks, labeling) required for the research. The dataset includes seven different attack types, i.e., Intrusion, Brute Force, Heartbleed, Botnet, DoS, DDoS, web attacks, and network infiltration. The infrastructure used for the simulation of the attacks consisted of 50 machines while the victim organization consisted of 5 departments, 420 machines, and 30 servers. The dataset contains a record of the network traffic and system logs of each machine and 80 attributes extracted from the recorded traffic using CICFlowMeter-V3 [30].

The dataset actually consists of several files. The files selected for investigation and the attack types covered by them are presented in Table 2. The preprocessing started with merging these files. The three stages of the preprocessing workflow are presented in the next three subsections.

### 3.2 Data cleaning

In order to create a proper dataset to train and test the model one has to preprocess the raw data. The preprocessing workflow starts with data cleaning, which usually includes deleting rows (records)

Table 2  
Selected datasets

File name	Attack types
Wednesday-14-02-2018 TrafficForML CICFlowMeter.csv	FTP-BruteForce SSH-BruteForce
Thursday-22-02-2018 TrafficForML CICFlowMeter.csv	Brute Force –Web Brute Force –XSS SQL Injection
Friday-23-02-2018 TrafficForML CICFlowMeter.csv	Brute Force –Web Brute Force –XSS SQL Injection

containing invalid or missing data, deleting one-valued columns (e.g. columns where all values are zero), as well as deleting features (columns) that are a-priory known to be irrelevant regarding the classification. The main steps are illustrated in Fig. 1. This could already achieve some dimensionality reduction, which can provide various advantages that will be discussed later.

For the current analysis, the time parameter is not needed, nor are the columns where all values are zero since they do not influence the output, i.e. the value of the last column. The data cleaning step resulted in 69 remaining columns after deleting 11 columns out of the original 80 ones. The next step was to delete rows containing invalid values. Therefore, first, the rows with the values *inf* and *-inf* were deleted, and then rows with negative values in the dataset were also deleted. With these operations, the cleaning of the dataset was completed.

### 3.3 Data transformation

In course of this research, the data transformation phase comprised three operations, i.e. the transformation of categorical data into numerical, normalization, and splitting of the dataset. To perform further calculations, all non-numerical elements of the dataset were converted into numbers. It was carried out in the case of each categorical column by assigning a number to each category. For example, each occurrence of the string "FTP-BruteForce" was replaced by the value 1. The order of the values was determined arbitrarily not taking into consideration any conceptual distance metrics mainly because later on the dataset was split into subsets containing only data corresponding to one attack type and normal traffic.

Data normalization is a common practice in machine learning, which consists of converting numeric columns to a common scale. In machine learning, some feature values are several times greater than others. Thus the features with higher values could dominate the learning process. However, it does not mean that these variables are more important in predicting the model output. Data normalization converts multiple scaled data to the same scale. After normalization, all variables have similar scale-related effects on the model, which improves the stability and performance of the learning algorithm. There are several types of normalization techniques. The simplest and most used type is the min-max scaling (1), which rescales a feature into the fixed range [0,1] by subtracting the minimum value of the feature ( $x_{min}$ ) from the current value ( $x$ ) and then dividing the result by the range.

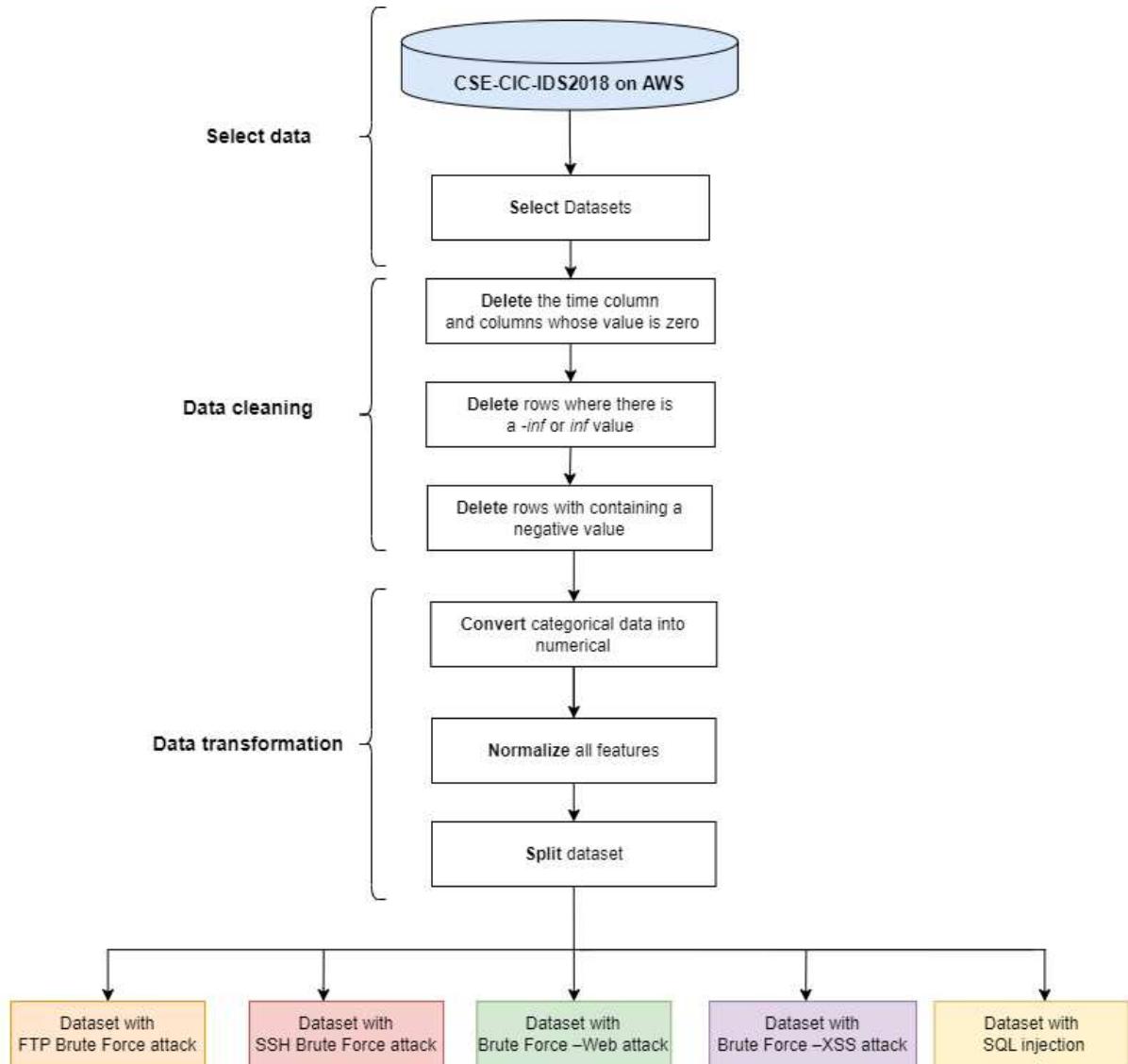


Fig. 1. Dataset preprocessing workflow

$$X_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

As a final step in the preprocessing, the selected datasets were split so that each resulting file contained only records corresponding to one attack type as well as records describing normal traffic. This operation resulted in five data files (see Fig. 1). Some of their features are presented in Table 3.

Table 3  
Datasets generated during preprocessing

<b>File name</b>	<b>Number of rows</b>	<b>Number of columns</b>
dataset-ftp.csv	857,162	69
dataset-ssh.csv	851,397	69
dataset-web.csv	2,085,515	69
dataset-xss.csv	2,085,134	69
dataset-sql.csv	2,084,991	69

### 3.4 Data reduction

The data reduction phase focuses on feature selection and dimensionality reduction, which can provide several advantages. One of the most important benefits is that many data mining algorithms work better when the number of dimensions - the number of attributes (columns) in the data - is smaller. This is partly because dimensionality reduction eliminates irrelevant attributes and reduces noise. Another advantage is that it can lead to a more comprehensible model because there will be fewer features in it. In addition, the reduced amount of data requires less storage space and less time for its processing.

There are many software tools that can be used to preprocess and analyze large datasets (e.g. Matlab, SPSS, Orange, Python). In the course of this research, we opted for the usage of the Python programming language because it is free of charge and its modules and libraries can produce fast and efficient results. The applied methods and the obtained results are presented in the next section.

## 4 Feature selection

Feature selection focuses on finding the most relevant attributes, which can be used to carry out an effective classification or prediction [31][32][33]. It contributes to the reduction of the dimensionality of the problem and so to the decrease of the resource requirements (storage, computation) as well as it can improve the performance of machine learning algorithms [34], i.e. faster training reduced over-fitting, and sometimes better prediction power. Although this approach may seem to lead to loss of information, this is not the case when redundant or irrelevant information is present. Redundant features are copies of most or all of the information found in one or more other attributes or they can be obtained as a combination of other features.

Irrelevant attributes contain almost no information that is useful for the data mining task to be performed. Redundant and irrelevant features can reduce the accuracy of the classification and the quality of the clusters discovered. While some irrelevant and redundant attributes can be removed immediately by common sense or professional knowledge, the selection of the best subset of attributes often requires a systematic approach.

The ideal approach to feature selection is trying all possible subsets of features as input to the data mining algorithm used and then selecting the subset that produced the best results. However, this technique would require an enormous amount of time and computational power. Therefore, several other methods have been developed for this purpose primarily based on statistical assumptions. There are three basic approaches to feature selection.

- *Wrapper methods*: the feature selection algorithm uses the learning method as a subroutine with the computational burden of invoking the learning algorithm to evaluate each subset of features. It finds the best feature set for a given type of machine learning algorithm.
- *Embedded methods*: the machine learning algorithm decides what attributes to use and what features to ignore.
- *Filter methods*: the features are selected before the data mining algorithm is run, using a method that is independent of the data mining task.

All the feature selection methods presented in the next subsections and used in course of this investigation belong to the group of filter methods. Their advantage is that their time complexity is the lowest among the three groups, and usually, after their application, the machine learning algorithm is less prone to over-fitting.

#### 4.1 Information Gain

$$H(Y) = - \sum p(y) \log_2(p(y)), \quad (2)$$

where  $p(y)$  is the marginal probability density function for the random variable  $Y$ . If the observed values of  $Y$  in the training dataset are partitioned according to the values of a second feature  $X$ , and the entropy of  $Y$  with respect to the partitions induced by  $X$  is less than the entropy of  $Y$  prior to partitioning, then there is a relationship between features  $Y$  and  $X$ . Thus the entropy of  $Y$  after observing  $X$  is

$$H(Y|X) = \sum p(x) \sum p(y|x) \log_2(p(y|x)), \quad (3)$$

where  $p(y|x)$  is the conditional probability of  $y$  given  $x$ . Given the entropy as a criterion of impurity in a dataset, one can define a measure reflecting additional information about  $Y$  provided by  $X$  that represents the amount by which the entropy of  $Y$  decreases. This measure, an indicator of the dependency between  $X$  and  $Y$ , is known as information gain (4).

$$IG(X, Y) = H(Y) - H(Y|X) = H(X) - H(X|Y). \quad (4)$$

IG is a symmetrical measure. The method provides an orderly classification of all the features, and then a threshold is required to select a certain number of them according to the order obtained. A weakness of the IG criterion is that it is biased in favor of features with more values even when they are not more informative [35].

#### 4.2 Gain Ratio

The information gain method prefers to select attributes having a large number of values, which led to the development of the feature selection method gain ratio (GR) which is a modification of the information gain aiming the decrease its bias. Originally developed for decision trees GR takes the number and size of the branches into account when choosing an attribute. It improves the evaluation given by information gain by taking into account the number of splits in the feature, i.e., how equally they are distributed [36]. GR reflects the relevance of each feature the higher its value is the higher the influence of the feature is. Gain ratio is calculated by formula (5)

$$GR(X, Y) = \frac{IG(X, Y)}{S \text{splitInfo}}, \quad (5)$$

$$S \text{splitInfo} = - \sum p(i) * \log_2(p(i)), \quad (6)$$

where  $p(i)$  represents the proportion of instances that fall into a particular split of the feature [37].

#### 4.3 Relief

The Relief method calculates a weight value for each feature ( $W_j$ = weight of feature ‘j’) that can be used to estimate the quality or relevance of the feature [38]. The weight vector is initialized with zero values, and it is updated using an iterative approach. The algorithm takes a random sample of  $m$  elements from the dataset. In the case of each instance of the sample ( $R_i$ ) it searches for the closest instance that belongs to the same class ( $H_i$ ) and for the closest instance that belongs to the other class ( $M_i$ ). Next, the value of each feature weight is updated by the formula (7).

$$W_j = W_j - \frac{D(R_{ij}, H_{ij})}{m} + \frac{D(R_{ij}, M_{ij})}{m}, \quad (7)$$

where  $X_{ij}$  is the  $j$ th feature of the  $i$ th instance (here  $X$  can be  $R$ ,  $H$ , or  $M$ ). The function  $D$  is defined as:

$$D(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise} \end{cases} \quad (8)$$

The shortcoming of Relief is that it does not identify redundant features, and it can be used only in the case of binary classification problems.

#### 4.4 Symmetric Uncertainty

Symmetric uncertainty (SU) can be used to calculate the rank of features for feature selection by calculating the relevance between the feature and the class label. A feature with a high SU value gets high importance [39]. SU is calculated by normalizing the double value of IG to the sum of the entropies of the two variables.

$$SU(X, Y) = \frac{2xIG(X, Y)}{H(X) + H(Y)}, \quad (9)$$

where  $H(X)$  and  $H(Y)$  are the entropy values of variables  $X$  and  $Y$ , while  $IG(X, Y)$  is the information gain related to the variables  $X$  and  $Y$ , respectively [40].

#### 4.5 Chi-Squared

The Chi-squared test is a statistical hypothesis test that measures divergence from the expected distribution if one assumes that the feature occurrence is actually independent of the class value [41]. The higher the value of chi-squared, the more relevant the feature with respect to the class is. Its calculation is based on the formula (10) [42].

$$\chi^2 = \sum_{i=1}^{n_I} \sum_{j=1}^{n_c} \frac{[A_{ij} - \frac{R_i B_j}{N}]^2}{\frac{R_i B_j}{N}}, \quad (10)$$

where  $n_I$  is the number of intervals,  $n_c$  is the number of classes,  $N$  is the total number of instances,  $A_{ij}$  the number of instances in the interval  $i$  and class  $j$ ,  $R_i$  denotes the number of instances in the interval  $i$ , and  $B_j$  the number of instances in class  $j$ . The Chi-squared test based evaluation was developed for discrete variables. Therefore in the case of continuous features before its application a discretization has to be carried out.

#### 4.6 ANOVA

Analysis of Variance (ANOVA) is a statistical analysis technique used to compare the means of multiple groups to determine if there is a significant difference between them. Similar to the Chi-squared approach a discretization is necessary before its usage [43]. The key idea of ANOVA is to compare the total variance of the data to the variation within the groups and the variation between the groups. The within-group sum of squares (SSW) (11) measures the variation within the groups. It is defined as

$$SSW = \sum_{i=1}^k [(n_i - 1) * SS(i)], \quad (11)$$

where  $n_i$  is the number of instances in group  $i$ , and  $SS(i)$  is the variance of group  $i$ .

The Sum of Squares between groups (SSB) (12) measures the variation between the means of the groups. It is defined as

$$SSB = K * \sum (x - \bar{x})^2 \quad (12)$$

where  $K$  is the number of groups,  $\bar{x}_i$  is the mean of group  $i$ , and  $\bar{x}$  is the mean of all instances.

The total sum of squares (SST) (13) is

$$SST = SSW + SSB \quad (13)$$

The null hypothesis of ANOVA is that all groups have the same mean, i.e., the values of the investigated feature do not have effect on the final class. The alternative hypothesis is that at least one group has a different mean. The null hypothesis is tested by the help of the F-ratio, which is the ratio of SSB to SSW (14). An F-ratio higher than a threshold value (called critical F-value) indicates that there is a significant difference between the means of the groups, and so the null hypothesis can be rejected.

$$F = SSB/SSW \quad (14)$$

The critical F-value is that value of the F-distribution, which is defined by the degrees of freedom for the numerator ( $df_{SSB} = K - 1$  and the degrees of freedom for the denumerator ( $df_{SSW} = N - K$ ). Here  $N$  is the number of instances.

#### 4.7 Feature selection results

The six feature selection methods presented in the previous section were applied for all five datasets using 30 university lab computers as well as the ELKH cloud services [44]. The feature selection workflow is presented in Fig. 2. All the necessary program elements were implemented in Python [45] [46]. Although several tasks were performed in parallel the whole process took more than two months.

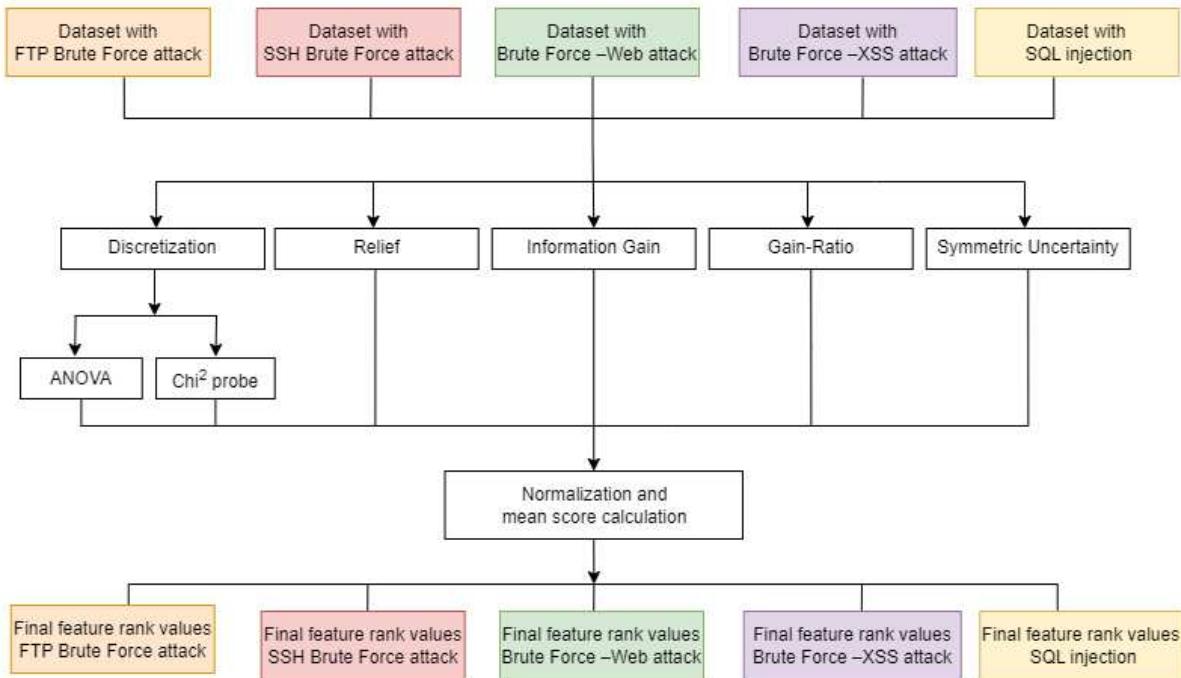


Fig. 2. Feature selection workflow

In the case of each dataset and each method, the feature score values obtained at the end of the feature selection process were normalized. Next, the final feature score was calculated in the case of each dataset separately as the mean of the normalized scores. The detailed results can be found in the Appendix in Tables A1 - A5.

Finally, five feature ranking threshold values were set starting from 0.35 and increasing by a step of 0.5. In the case of each value, we selected the features whose score was higher than the threshold. The results are presented in Table 4. In the case of each attack type a separate list of relevant features was identified. Each feature is represented by its ordinal value. Each row of the table contains those features

whose score was greater or equal to the threshold given in the first cell. Starting from the second column each column represents an attack type.

Having the selected feature groups we continued our investigation by applying different classification methods that will be presented in the following section.

Table 4  
Feature selection based on threshold values

<b>Ranking threshold</b>	<b>FTP</b>	<b>SSH</b>	<b>WEB</b>	<b>XSS</b>	<b>SQL</b>
0.35	02, 17, 19, 35, 00, 44, 56, 59	00, 02, 17, 19, 57, 56, 59	16, 20, 10, 49, 66, 67, 35, 38, 56, 64, 34, 27, 07, 09, 11, 14, 15, 50, 25, 60, 62, 02, 17, 19, 37, 63, 06, 33, 55, 18, 58, 04, 05, 53, 54, 03, 21, 22, 23, 24, 52, 32, 65, 57	25, 27, 16, 40, 02, 05, 17, 19, 34, 53, 06, 18, 55, 21, 22, 23, 24, 51, 13, 39, 57, 37, 56, 33, 32, 03, 11, 52, 04, 54, 58	39, 43, 47, 10, 15, 05, 26, 53, 56, 25, 02, 17, 19, 35, 16, 18, 27, 28, 34, 06, 23, 30, 55, 29, 21, 22, 24, 57, 37, 11, 14
0.40	44, 56, 59	56, 59	34, 27, 07, 09, 11, 14, 15, 50, 25, 60, 62, 02, 17, 19, 37, 63, 06, 33, 55, 18, 58, 04, 05, 53, 54, 03, 21, 22, 23, 24, 52, 32, 65, 57	02, 05, 17, 19, 34, 53, 06, 18, 55, 21, 22, 23, 24, 51, 13, 39, 57, 37, 56, 33, 32, 03, 11, 52, 04, 54, 58	05, 26, 53, 56, 25, 02, 17, 19, 35, 16, 18, 27, 28, 34, 06, 23, 30, 55, 29, 21, 22, 24, 57, 37, 11, 14
0.45	56, 59	56, 59	02, 17, 19, 37, 63, 06, 33, 55, 18, 58, 04, 05, 53, 54, 03, 21, 22, 23, 24, 52, 32, 65, 57	37, 56, 33, 32, 03, 11, 52, 04, 54, 58	06, 23, 30, 55, 29, 21, 22, 24, 57, 37, 11, 14
0.50	56, 59	59	03, 21, 22, 23, 24, 52, 32, 65, 57	03, 11, 52, 04, 54, 58	57, 37, 11, 14
0.55	56, 59	59	57	58	11, 14

## 5 Classification algorithms

Classification methods are used to predict the class of an object instance based on a feature vector. Machine learning-based classification algorithms build models that can learn from labeled datasets and use them to predict the class of new, unseen data points. In this investigation, we used five different classification algorithms representing four main classification groups. These groups are linear models, probabilistic models, tree-based models, and kernel-based models.

Linear models are represented by the Logistic Regression method, which models the probability of a binary outcome using a sigmoid function.

Probabilistic models are represented by the Naive Bayes model, which assumes that the features are independent given the class and uses Bayes' theorem to compute the posterior probabilities of each class.

Tree-based models are represented by two methods: the Decision Tree method, a non-parametric model that recursively partitions the feature space into a tree structure, and the Random Forest method, an ensemble model that uses multiple decision trees and aggregates their predictions to improve performance.

Kernel-based models are represented by the Support Vector Machine (SVM) method, which maps the input data into a high-dimensional feature space and finds a hyperplane that maximally separates the classes.

The following subsections provide a brief description of the classification algorithms mentioned above.

### 5.1 Logistic Regression

Logistic Regression (LR) is a linear classification method that estimates the probability of a certain instance belonging to a class (e.g. attack). LR belongs to the family of linear methods and is an alternative to discriminant analysis. Its application prerequisites are less strict than those of discriminant analysis [47]. The key idea of Logistic Regression is to calculate a linear combination (see eq. (15)) of the feature values  $X = [x_1, x_2, \dots, x_n]$  for each observation instance, using a coefficient vector  $A = [a_0, a_1, \dots, a_n]$  that is determined during the classifier's training.

$$Z(x) = a_0 + a_1 * x_1 + \dots + a_n * x_n, \quad (15)$$

To determine the probability of belonging to the attack class (class 1) Logistic Regression applies a sigmoid function to  $Z$  that maps  $Z$  to the  $[0, 1]$  interval.

$$h(z) = \frac{1}{1 + e^{-z}}, \quad (16)$$

Finally, the classifier decides the final class of the observation by comparing the resulting probability value  $h$  to a threshold value  $h_{tr}$ , as shown in equation 17.

$$C(h) = \begin{cases} 1 & \text{if } h > h_{tr} \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

The threshold value  $h_{tr}$  is determined during the training phase of the classifier.

### 5.2 Naive Bayes

The Naive Bayes classification method is based on Bayes' theorem of conditional probability. It determines the predicted class of an  $X = [x_1, x_2, \dots, x_n]$  observation using the formula

The Naive Bayes classification method is based on Bayes' theorem of conditional probability. It predicts the class of an observation  $X = [x_1, x_2, \dots, x_n]$  using the formula (18)

$$C(x) = \arg \max_{c_j \in C} P(c_j) \prod_{i=1}^n P(x_i | c_j) \quad (18)$$

where  $c_j$  is the  $j$ th class,  $x_i$  is the value of the  $i$ th feature,  $n$  is the number of features,  $P(c_j)$  is the prior probability of class  $j$ , and  $P(x_i|c_j)$  is the conditional probability of the value of  $x_i$  given class  $c_j$ . The prior probability  $P(c_j)$  is estimated by the relative frequency of class  $j$  in the training sample.

In the case of categorical features  $P(x_i|c_j)$  is estimated by the relative frequency of the value  $x_i$  among the training sample elements that belong to class  $c_j$ . In the case of continuous features  $P(x_i|c_j)$  is estimated by the value of the probability density function calculated for  $x_i$  taking into consideration the training sample elements that belong to class  $c_j$

$$P(x_i|c_j) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}}, \quad (19)$$

where  $\mu_i$  is the mean value and  $\sigma_i$  is the standard deviation of the  $i$ th feature among the training sample elements taken into consideration.

### *5.3 Support Vector Machine*

Support Vector Machine (SVM) [48] is a statistically based supervised classification technique that can be used to efficiently handle high-dimensional data. It creates a multi-dimensional hyperplane that separates the two classes in the case of binary classification problems. Multiclass problems are reduced to multiple binary classification problems.

If no simple linear separation can be carried out it transforms the data by using so-called kernel functions that calculate the hyperplane in a higher dimension. The nonlinearity of the hyperplane can also be tuned with the help of the regularization and the gamma parameters. The value of the regularization parameter describes how much one wants to avoid misclassification in the case of training instances. A high value could result in a more complex hyperplane with a small amount of wrongly classified data points if any.

In the case of high gamma values, only training instances close to the hyperplane will be considered in the course of its definition.

### *5.4 Decision Tree*

Decision trees offer an easy-to-interpret and visualize tool for classification. They make the decision based on rules inferred from the feature values of the training sample. Each leaf of the tree corresponds to a class label. At each node, only one feature is taken into consideration and no root-to-leaf path contains twice the same feature. A Classification tree may also provide a confidence measure regarding the quality of the classification. The tree is built up from the training sample in a recursive manner [49]. It is an iterative process whereby data is partitioned into partitions and then further partitioned on each branch. The features used at different nodes are selected using statistical methods like Information Gain or Gini Index. If all features are already used and the remaining sample contains instances belonging to more than one class a leaf is created and its class will be decided using a majority vote.

### *5.5 Random forest*

The Random Forest (RF) method [50] was developed to overcome a shortcoming of Decision Trees, i.e., having the tendency to overfit the sample data. RF mitigates this problem by using a statistical

technique called bootstrapping, which generates multiple models and combines their results to make a final decision. The main idea behind RF is that by aggregating the predictions of multiple classifiers, the impact of individual errors can be minimized.

In course of Bootstrapping several smaller samples are drawn from the training dataset randomly with replacement. Each sample is used to train a separate classifier. Thus when classifying a new observation its final class prediction is made by aggregating the results given by the individual models. Usually, it is done by applying a majority voting solution.

## 6 Training the classifiers - Experimental results

All five datasets contained a very large number of instances (see Table 3). Therefore when creating the training and test samples only a part of the original data was used. The steps of the training-test sample construction are presented in Fig. 3.

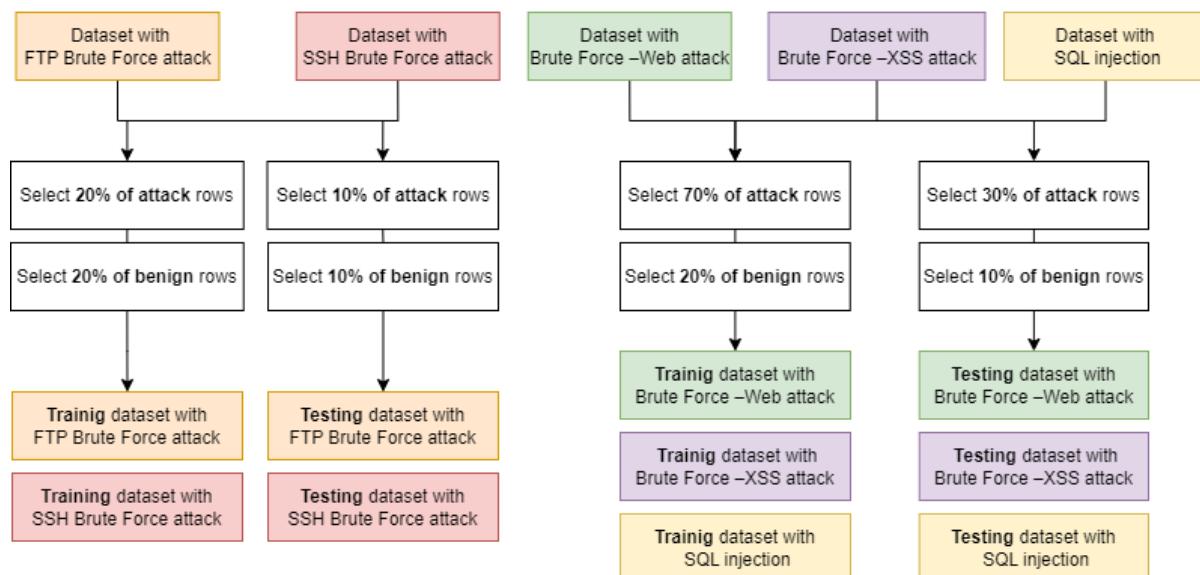


Fig. 3. Creation of training and testing datasets

Both in the case of the FTP and SSH Brute Force attacks training samples contained 20% of the original instances. We applied stratified sampling to ensure that each class (attack and benign traffic) is represented in the sample. Thus the resulting collection of records contained 20% of the attack rows and 20% of the rows describing benign traffic. The sampling was carried out without replacement. The test samples were created in a similar way by choosing records from the remaining datasets so that the resulting collection of data points represented 10% of the original ones. The resulting record numbers are shown in Tables 5 and 6, respectively.

In the case of the Brute Force Web, Brute Force XSS, and SQL Injection attack types the selection process was slightly different owing to the fact that the number of records describing malicious traffic was very small. Therefore in each case, the collection of attack rows was split into two parts, i.e., 70%

was used for training and the remaining 30% for test purposes. Next, the training samples were created by adding 20% of the data points belonging to the benign traffic. Finally, the test samples were compiled by adding 10% of the benign traffic record to the attack rows allocated for test purposes. The resulting record numbers are shown in Tables 5 and 6, respectively.

Table 5  
Training datasets

File name	Number of rows	Number of columns
dataset-ftp-tr.csv	171,433	69
dataset-ssh-tr.csv	170,280	69
dataset-web-tr.csv	417,332	69
dataset-xss-tr.csv	417,218	69
dataset-sql-tr.csv	417,042	69

Table 6  
Test datasets

File name	Number of rows	Number of columns
dataset-ftp-ts.csv	85,716	69
dataset-ssh-ts.csv	85,140	69
dataset-web-ts.csv	208,636	69
dataset-xss-ts.csv	208,598	69
dataset-sql-ts.csv	208,517	69

The training and testing of the five classifiers was carried out in Orange 3.34, which is an open-source data visualization, machine learning, and data mining toolkit. It offers a visual programming front-end for interactive data visualization and exploratory, quick qualitative data analysis. Its components are called widgets and they range from simple data visualization, subset selection, and preprocessing to empirical evaluation of learning algorithms and predictive modeling. Visual programming is implemented through an interface in which workflows are created by linking predefined or user-designed widgets, while advanced users can use Orange as a Python library for data manipulation and widget alteration. Orange uses common Python open-source libraries for scientific computing, such as numpy, scipy, and scikit-learn, while its graphical user interface operates within the cross-platform Qt framework.

The classifier training and testing workflow used in course of the investigation is shown in Figure 4. It was carried out separately for each attack type and for each relevant feature collection. For example, in the case of the FTP Brute Force attack and the 0.40 ranking threshold value three features (44, 56, and 59) were supposed to play a significant role. Thus in total 21 workflow executions were necessary and 105 classifiers were trained.

All the classifiers were evaluated against the training and test samples using four measures, i.e., accuracy, precision, recall, and F1. The detailed results can be found in the Appendix in Tables A6 - A10.

In the case of FTP attacks each of the classifiers performed quite well having high accuracy values and the highest possible recall rates for almost all of the feature subset-classifier type pairs. Except for the case of the logistic regression-based classifier, the accuracy against the training dataset usually

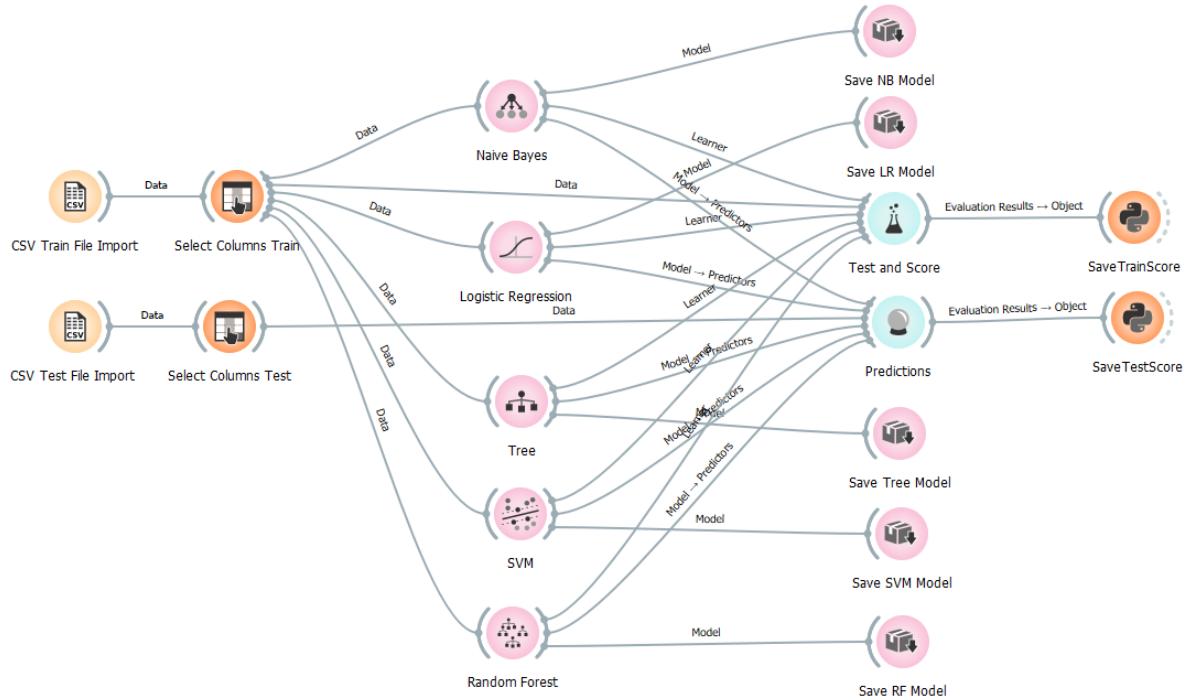


Fig. 4. Classifier training and testing workflow

improved with increasing the number of selected features (see Fig 5). However, when evaluating the classifiers with the test dataset a slight decay in accuracy performance could be measured in the case of Naive Bayes and Random Forest classifiers as well (see Fig 6).

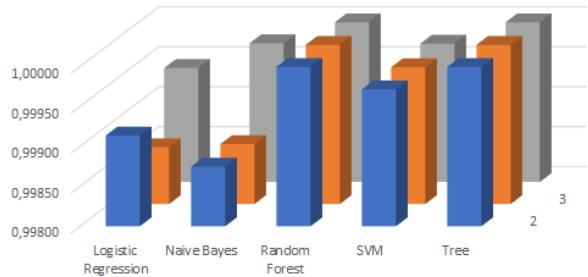


Fig. 5. Accuracy values in case of the FTP attack and the training dataset

The classifiers also exhibited strong performance against SSH attacks, with high accuracy observed for all feature subset-classifier pairs. Increasing the number of selected features led to improved accuracy against the training dataset, except in the case of the SVM-based classifier (see Fig 7). When evaluating the classifiers against the test dataset revealed a very similar behavior (see Fig 8).

In the case of Web attacks, the SVM classifier provided a low accuracy rate compared to the others both in the case of the training (see Fig. 9) and test (see Fig. 10) datasets. However, Logistic regression,

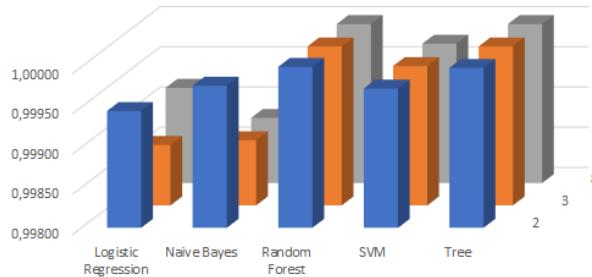


Fig. 6. Accuracy values in case of the FTP attack and the test dataset

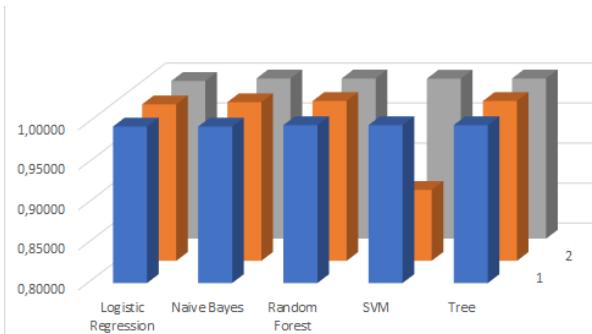


Fig. 7. Accuracy values in case of the SSH attack and the training dataset

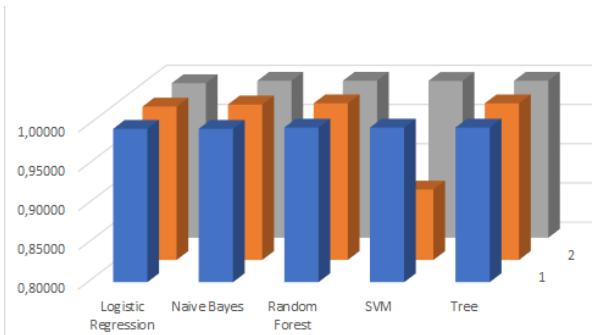


Fig. 8. Accuracy values in case of the SSH attack and the test dataset

Random Forest, and Decision Tree based classifiers were able to successfully predict the nature of the traffic with a very high accuracy rate. Although the Naive Bayes model in the case of 23 and 34 selected features showed a declining performance its results were not too much fallen behind.

Among the classifiers tested for XSS attacks, the SVM classifier had a low accuracy rate for both the training dataset (see Fig.11) and test dataset (see Fig.12) in comparison to others. The Logistic Regression, Random Forest, and Decision Tree based classifiers performed exceptionally well with a very high accuracy rate. While the Naive Bayes model showed declining performance in the case of 27 and 31 selected features, its results were still competitive.

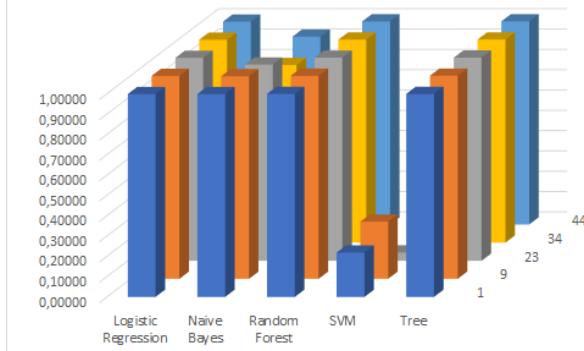


Fig. 9. Accuracy values in case of the WEB attack and the training dataset

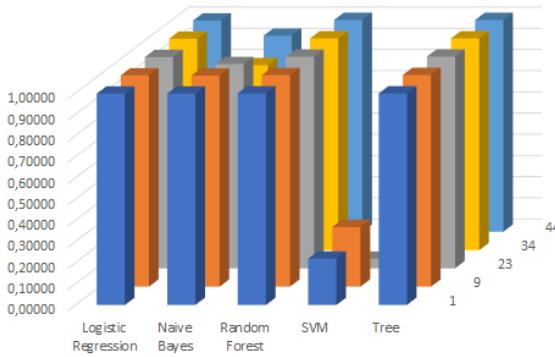


Fig. 10. Accuracy values in case of the WEB attack and the test dataset

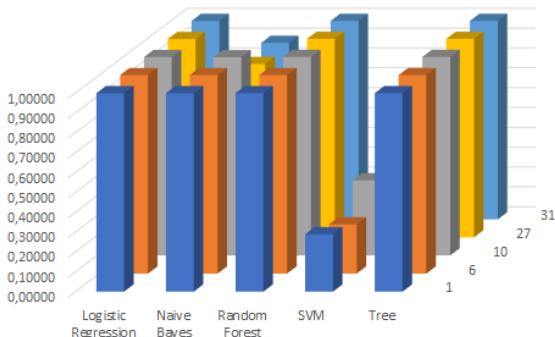


Fig. 11. Accuracy values in case of the XSS attack and the training dataset

For SQL Injection attacks, four of the five classifier models demonstrated high accuracy rates against both the training dataset (see Fig. 13) and test dataset (see Fig. 14). The Naive Bayes model was the only exception, showing slightly declining performance when using 26 and 31 selected features, but still having accuracy values over 0.9.

The best-performing classifiers along with ranking threshold values and selected feature number as well as the evaluation results are highlighted in Table 7

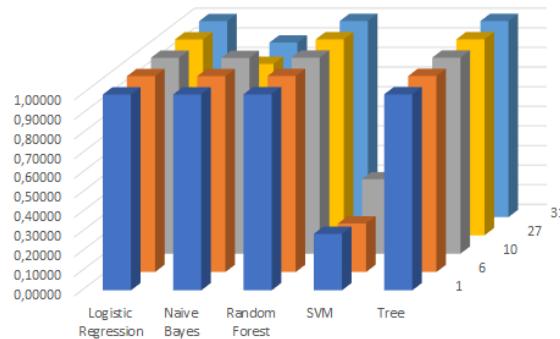


Fig. 12. Accuracy values in case of the XSS attack and the test dataset

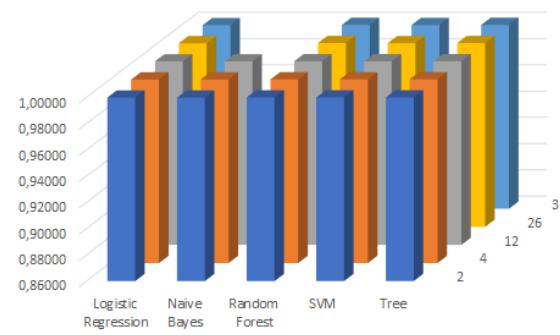


Fig. 13. Accuracy values in case of the SQL attack and the training dataset

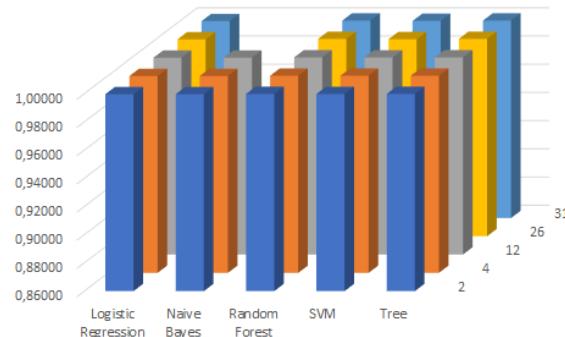


Fig. 14. Accuracy values in case of the SQL attack and the test dataset

## 7 Conclusions

In course of the investigation reported this paper, six feature evaluation techniques were performed on five datasets after completing the data cleaning and transformation steps. Each dataset comprised records that described two types of traffic cases: benign and attack, and featured 69 attributes. An average score was calculated after normalization and used to rank individual features. Next, six ranking thresholds

Table 7  
Best performing classifiers

Attack Type	Ranking Threshold	Features	Classifier	Accuracy	Precision	Recall	F1
FTP	0.35	8	Random Forest	1.00000	1.00000	1.00000	1.00000
SSH	0.35	7	Random Forest	0.99999	0.99997	1.00000	0.99999
WEB	0.35	44	Tree	0.99994	0.98997	0.96890	0.97932
XSS	0.45	10	Random Forest	0.99999	1.00000	0.97391	0.98678
SQL	0.40	26	Tree	0.99999	1.00000	0.95402	0.97647

were defined, which led to the selection of several relevant feature collections for each attack type. The number of included attributes varied widely, ranging from 1 (SSH) to 44 (Web).

Next, five classifier models were trained for each collection using the Orange software tool, and their performance was evaluated against the train and test datasets using four classification metrics. It was observed that, in some cases, accuracy improved slightly when increasing the number of features. However, excellent results were achieved in most cases, even with a low number of attributes. Table 7, which shows the best-performing classifiers for each attack type, clearly indicates that no general threshold can be set for the feature scores. The results suggest that tree-type classification algorithms represent the most appropriate solution for the investigated attack types when feature selection followed the presented workflow.

The methodology utilized in the current investigation can be applied to other scenarios involving high-dimensional data, such as clustering (e.g. [51][52]), object identification [53], classification [54], indoor localization [55], or technology optimization [56].

Further research will focus on the investigation of the suitability of different aggregation techniques (e.g. [57][58]) that could replace the average score in feature relevance calculation. Furthermore the applicability of further computational intelligence methods.

## Acknowledgements

On behalf of the project we are grateful for the possibility to use ELKH Cloud [44]; (<https://science-cloud.hu/>), which helped us achieve the results published in this paper.

This research was supported by 2020-1.1.2-PIACI-KFI-2020-00062 "Development of an industrial 4.0 modular industrial packaging machine with integrated data analysis and optimization based on artificial intelligence, error analysis". The Hungarian Government supports the Project and is co-financed by the European Social Fund.

## References

- [1] Z.C. L. Göcs Johanyák, Survey On Intrusion Detection Systems, in: *7th International Scientific and Expert Conference TEAM 2015 Technique, Education, Agriculture & Management*, 2015.
- [2] Z.C. L. Göcs Johanyák and S. Kovács, Review of Anomaly-Based IDS algorithms, in: *8th International Scientific and Expert Conference TEAM 2016 Technique, Education, Agriculture & Management.*, 2016.

- [3] D. Stiawan, M.Y.B. Idris, A.M. Bamhdi, R. Budiarto et al., CICIDS-2017 dataset feature analysis with information gain for anomaly detection, *IEEE Access* **8** (2020), 132911–132921. doi:10.1109/ACCESS.2020.3009843.
- [4] M.A. Rahman, A.T. Asyhari, O.W. Wen, H. Ajra, Y. Ahmed and F. Anwar, Effective combining of feature selection techniques for machine learning-enabled IoT intrusion detection, *Multimedia Tools and Applications* **80**(20) (2021), 31381–31399. doi:<http://dx.doi.org/10.1007/s11042-021-10567-y>.
- [5] A. Javadpour, S.K. Abharian and G. Wang, Feature selection and intrusion detection in cloud environment based on machine learning algorithms, in: *2017 IEEE international symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ISPA/IUCC)*, IEEE, 2017, pp. 1417–1421. doi:<https://doi.org/10.1109/ISPA/IUCC.2017.00215>.
- [6] K.A. Taher, B.M.Y. Jisan and M.M. Rahman, Network intrusion detection using supervised machine learning technique with feature selection, in: *2019 International conference on robotics, electrical and signal processing techniques (ICREST)*, IEEE, 2019, pp. 643–646. doi:<https://doi.org/10.1109/icrest.2019.8644161>.
- [7] G. Kocher and G. Kumar, Analysis of machine learning algorithms with feature selection for intrusion detection using UNSW-NB15 dataset, *Available at SSRN* 3784406 (2021). doi:<https://doi.org/10.5121/ijnsa.2021.13102>.
- [8] M. Alkasassbeh, An empirical evaluation for the intrusion detection features based on machine learning and feature selection methods, *arXiv preprint arXiv:1712.09623* (2017). doi:<https://doi.org/10.48550/arXiv.1712.09623>.
- [9] I.S. Thaseen and C.A. Kumar, Intrusion detection model using fusion of chi-square feature selection and multi class SVM, *Journal of King Saud University-Computer and Information Sciences* **29**(4) (2017), 462–472. doi:10.1016/j.jksuci.2015.12.004.
- [10] J.B. Awotunde, C. Chakraborty and A.E. Adeniyi, Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection, *Wireless communications and mobile computing* **2021** (2021). doi:10.1155/2021/7154587.
- [11] H.P.S. Sasan and M. Sharma, Intrusion detection using feature selection and machine learning algorithm with misuse detection, *International Journal of Computer Science and Information Technology* **8**(1) (2016), 17–25. doi:10.5121/ijcsit.2016.8102.
- [12] S.K. Biswas et al., Intrusion detection using machine learning: A comparison study, *International Journal of pure and applied mathematics* **118**(19) (2018), 101–114.
- [13] A. Ali, S. Shaukat, M. Tayyab, M.A. Khan, J.S. Khan, J. Ahmad et al., Network intrusion detection leveraging machine learning and feature selection, in: *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, IEEE, 2020, pp. 49–53. doi:10.1109/honet50430.2020.9322813.
- [14] H. Malhotra, P. Sharma et al., Intrusion detection using machine learning and feature selection, *International Journal of Computer Network and Information security* **11**(4) (2019), 43. doi:10.5815/ijcnis.2019.04.06.
- [15] S. Krishnaveni, S. Sivamohan, S. Sridhar and S. Prabakaran, Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing, *Cluster Computing* **24**(3) (2021), 1761–1779. doi:<https://doi.org/10.1007/s10586-020-03222-y>.
- [16] K. Kumar and J.S. Battu, Network intrusion detection with feature selection techniques using machine-learning algorithms, *International Journal of Computer Applications* **150**(12) (2016). doi:10.5120/ijca2016910764.
- [17] A. Pattawaro and C. Polprasert, Anomaly-based network intrusion detection system through feature selection and hybrid machine learning technique, in: *2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE)*, IEEE, 2018, pp. 1–6. doi:10.1109/ictke.2018.8612331.
- [18] T. Ahmad and M.N. Aziz, Data preprocessing and feature selection for machine learning intrusion detection systems, *ICIC Express Lett* **13**(2) (2019), 93–101. doi:10.24507/icicel.13.02.93.
- [19] P. Aggarwal and S.K. Sharma, Analysis of KDD dataset attributes-class wise for intrusion detection, *Procedia Computer Science* **57** (2015), 842–851. doi:10.1016/j.procs.2015.07.490.
- [20] R.B. Basnet, R. Shash, C. Johnson, L. Walgren and T. Doleck, Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks., *J. Internet Serv. Inf. Secur.* **9**(4) (2019), 1–17.
- [21] I. Sharafaldin, A.H. Lashkari and A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., *ICISSP* **1** (2018), 108–116. doi:10.5220/0006639801080116.
- [22] M. Chen, A.A. Ghorbani et al., A survey on user profiling model for anomaly detection in cyberspace, *Journal of Cyber Security and Mobility* **8**(1) (2019), 75–112. doi:10.13052/jcsm2245-1439.814.
- [23] I. Sharafaldin, A. Gharib, A.H. Lashkari and A.A. Ghorbani, Towards a reliable intrusion detection benchmark dataset, *Software Networking* **2018**(1) (2018), 177–200. doi:10.13052/jsn2445-9739.2017.009.
- [24] M. Ring, S. Wunderlich, D. Grüdl, D. Landes and A. Hotho, Creation of flow-based data sets for intrusion detection, *Journal of Information Warfare* **16**(4) (2017), 41–54.
- [25] M. Ring, S. Wunderlich, D. Grüdl, D. Landes and A. Hotho, Flow-based benchmark data sets for intrusion detection, in: *Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI*, 2017, pp. 361–369.

- [26] N. Moustafa and J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: *2015 military communications and information systems conference (MilCIS)*, IEEE, 2015, pp. 1–6. doi:10.1109/MilCIS.2015.7348942.
- [27] N. Moustafa and J. Slay, The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, *Information Security Journal: A Global Perspective* **25**(1–3) (2016), 18–31. doi:10.1080/19393555.2015.1125974.
- [28] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro and R. Therón, UGR ‘16: A new dataset for the evaluation of cyclostationarity-based network IDSs, *Computers & Security* **73** (2018), 411–424. doi:10.1016/j.cose.2017.11.004.
- [29] M.H. Bhuyan, D.K. Bhattacharyya and J.K. Kalita, Towards Generating Real-life Datasets for Network Intrusion Detection., *Int. J. Netw. Secur.* **17**(6) (2015), 683–701.
- [30] A.H. Lashkari, G. Draper-Gil, M.S.I. Mamun, A.A. Ghorbani et al., Characterization of tor traffic using time based features. (2017), 253–262. doi:10.5220/0006105602530262.
- [31] K. Muhi and Z.C. Johanyák, Dimensionality reduction methods used in Machine Learning, *Műszaki Tudományos Közlemények* **13**(1) (2020), 148–151. doi:10.33894/mtk-2020.13.27.
- [32] Z.J. Viharos, K.B. Kis, Á. Fodor and M.I. Büki, Adaptive, hybrid feature selection (AHFS), *Pattern Recognition* **116** (2021), 107932. doi:10.1016/j.patcog.2021.107932.
- [33] T. Dobján and E.D. Antal, Modern feature extraction methods and learning algorithms in the field of industrial acoustic signal processing, in: *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, IEEE, 2017, pp. 000065–000070. doi:10.1109/sisy.2017.8080589.
- [34] N.S. Chauhan, Decision Tree Algorithm—Explained, 2020, KDnuggets.[Online]. Available: [Accessed 16 April 2021].
- [35] A.G. Karegowda, A. Manjunath and M. Jayaram, Comparative study of attribute selection using gain ratio and correlation based feature selection, *International Journal of Information Technology and Knowledge Management* **2**(2) (2010), 271–277.
- [36] R.P. Priyadarsini, M. Valarmathi and S. Sivakumari, Gain ratio based feature selection method for privacy preservation, *ICTACT Journal on soft computing* **1**(4) (2011), 201–205. doi:10.21917/ijsc.2011.0031.
- [37] S.J. Pasha and E.S. Mohamed, Ensemble gain ratio feature selection (EGFS) model with machine learning and data mining algorithms for disease risk prediction, in: *2020 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, 2020, pp. 590–596. doi:10.1109/ICICT48043.2020.9112406.
- [38] R.J. Urbanowicz, M. Meeker, W. La Cava, R.S. Olson and J.H. Moore, Relief-based feature selection: Introduction and review, *Journal of biomedical informatics* **85** (2018), 189–203. doi:10.1016/j.jbi.2018.07.014.
- [39] B. Singh, N. Kushwaha, O.P. Vyas et al., A feature subset selection technique for high dimensional data using symmetric uncertainty, *Journal of Data Analysis and Information Processing* **2**(04) (2014), 95. doi:10.4236/jdaip.2014.24012.
- [40] S. Bakhshandeh, R. Azmi and M. Teshnehab, Symmetric uncertainty class-feature association map for feature selection in microarray dataset, *International Journal of Machine Learning and Cybernetics* **11**(1) (2020), 15–32. doi:10.1007/s13042-019-00932-7.
- [41] G. Forman et al., An extensive empirical study of feature selection metrics for text classification., *J. Mach. Learn. Res.* **3**(Mar) (2003), 1289–1305.
- [42] V. Bolón-Canedo, N. Sánchez-Maróño and A. Alonso-Betanzos, *Feature selection for high-dimensional data*, Springer, 2015. ISBN 978-3-319-21857-1. doi:10.1007/978-3-319-21858-8.
- [43] M. Kumar, N.K. Rath, A. Swain and S.K. Rath, Feature selection and classification of microarray data using MapReduce based ANOVA and K-nearest neighbor, *Procedia Computer Science* **54** (2015), 301–310. doi:10.1016/j.procs.2015.06.035.
- [44] M. Héder, E. Rigó, D. Medgyesi, R. Lovas, S. Tenczer, F. Török, A. Farkas, M. Emődi, J. Kadlecik, G. Mező, Á. Pintér and P. Kacsuk, The Past, Present and Future of the ELKH Cloud, *Információs Társadalom* **22**(2) (2022), 128. doi:10.22503/inftars.xxi.2022.2.8.
- [45] J. Bernard and J. Bernard, Python data analysis with pandas, *Python Recipes Handbook: A Problem-Solution Approach* (2016), 37–48.
- [46] W. McKinney et al., Data structures for statistical computing in python, in: *Proceedings of the 9th Python in Science Conference*, Vol. 445, Austin, TX, 2010, pp. 51–56.
- [47] M. Maalouf, Logistic regression in data analysis: an overview, *International Journal of Data Analysis Techniques and Strategies* **3**(3) (2011), 281–299. doi:10.1504/IJDATS.2011.041335.
- [48] I. Steinwart and A. Christmann, *Support vector machines*, 1st ed edn, Information science and statistics, Springer, New York, 2008. ISBN 978-0-387-77241-7 978-0-387-77242-4.
- [49] B. Charbuty and A. Abdulazeez, Classification based on decision tree algorithm for machine learning, *Journal of Applied Science and Technology Trends* **2**(01) (2021), 20–28. doi:10.38094/jastt20165.
- [50] L. Breiman, Random forests, *Machine learning* **45** (2001), 5–32. doi:10.1023/A:1010933404324.

- [51] S. Blažič and I. Škrjanc, Incremental fuzzy c-regression clustering from streaming data for local-model-network identification, *IEEE transactions on fuzzy systems* **28**(4) (2019), 758–767. doi:10.1109/TFUZZ.2019.2916036.
- [52] I.-D. Borlea, R.-E. Precup, A.-B. Borlea and D. Iercan, A unified form of fuzzy C-means and K-means algorithms and its partitional implementation, *Knowledge-Based Systems* **214** (2021), 106731. doi:10.1016/j.knosys.2020.106731.
- [53] J. Hvizdoš, J. Vaščák and A. Brezina, Object identification and localization by smart floors, in: *2015 IEEE 19th International Conference on Intelligent Engineering Systems (INES)*, IEEE, 2015, pp. 113–117. doi:10.1109/INES.2015.7329649.
- [54] S. Duer, L. Pokoradi, D. Bernatowicz and R. Duer, Classification of elements in the diagnostic model of a technical object for building an expert knowledge base, *Journal of Mechanical and Energy Engineering* **Vol. 1, No 1** (2017), 71–78.
- [55] D. Vincze and M. Niitsuma, What-You-See-Is-What-You-Get Indoor Localization for Physical Human-Robot Interaction Experiments, in: *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2022, pp. 909–914. doi:10.1109/AIM52237.2022.9863359.
- [56] M. Babič, I. Karabegović, S.I. Martinčić and G. Varga, New method of sequences spiral hybrid using machine learning systems and its application to engineering, in: *New Technologies, Development and Application 4*, Springer, 2019, pp. 227–237. doi:10.1007/978-3-319-90893-9\_28.
- [57] F. Lilik, Á. Bükovics and L.T. Kóczy, Fuzzy Inference System-like Aggregation Operator for Fuzzy Signatures, in: *Computational Intelligence and Mathematics for Tackling Complex Problems 4*, Springer, 2022, pp. 93–101. doi:10.1007/978-3-031-07707-4\_12.
- [58] E. Tóth-Laufer and M. Takács, The effect of aggregation and defuzzification method selection on the risk level calculation, in: *2012 IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, IEEE, 2012, pp. 131–136. doi:10.1109/SAMI.2012.6208943.