INFORMATICA, 2021, Vol. 32, No. 3, 441–475

DOI: https://doi.org/10.15388/21-INFOR457

Study of Multi-Class Classification Algorithms' Performance on Highly Imbalanced Network Intrusion Datasets

Viktoras BULAVAS[1,*], Virginijus MARCINKEVIČIUS[1], Jacek RUMIŃSKI[2]

[1] *Institute of Data Science and Digital Technologies, Vilnius University, Akademijos str. 4, LT-08663 Vilnius, Lithuania*

[2] *Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, 11/12 Gabriela Narutowicza, 80-233 Gdańsk, Poland*

e-mail: viktoras.bulavas@itpc.vu.lt, virginijus.marcinkevicius@mif.vu.lt, jacek.ruminski@pg.edu.pl

Received: March 2021; accepted: July 2021

Abstract. This paper is devoted to the problem of class imbalance in machine learning, focusing on the intrusion detection of rare classes in computer networks. The problem of class imbalance occurs when one class heavily outnumbers examples from the other classes. In this paper, we are particularly interested in classifiers, as pattern recognition and anomaly detection could be solved as a classification problem. As still a major part of data network traffic of any organization network is benign, and malignant traffic is rare, researchers therefore have to deal with a class imbalance problem. Substantial research has been undertaken in order to identify these methods or data features that allow to accurately identify these attacks. But the usual tactic to deal with the imbalance class problem is to label all malignant traffic as one class and then solve the binary classification problem. In this paper, however, we choose not to group or to drop rare classes but instead investigate what could be done in order to achieve good multi-class classification efficiency. Rare class records were up-sampled using SMOTE method (Chawla *et al.*, 2002) to a preset ratio targets. Experiments with the 3 network traffic datasets, namely CIC-IDS2017, CSE-CIC-IDS2018 (Sharafaldin *et al.*, 2018) and LITNET-2020 (Damasevicius *et al.*, 2020) were performed aiming to achieve reliable recognition of rare malignant classes available in these datasets.

Popular machine learning algorithms were chosen for comparison of their readiness to support rare class detection. Related algorithm hyper parameters were tuned within a wide range of values, different data feature selection methods were used and tests were executed with and without oversampling to test the multiple class problem classification performance of rare classes.

Machine learning algorithms ranking based on *Precision*, *Balanced Accuracy Score*, $\bar{G}$, and prediction error *Bias and Variance decomposition*, show that decision tree ensembles (*Adaboost, Random Forest Trees and Gradient Boosting Classifier*) performed best on the network intrusion datasets used in this research.

Key words: network intrusion detection, multi-class classification, imbalanced learning, bias and variance decomposition, SMOTE.

*Corresponding author.

1. Introduction

Detection of intrusions into networks, information systems or workstations, as well as detection of malware and unauthorized activities of individuals, have emerged into a global challenge. A part of cybernetic defence challenges is addressed by optimizing the intrusion detection systems (IDS). There are three methods of intrusion detection (Koch, 2011): known pattern recognition (signature-based), anomaly based detection, and a hybrid of the previous two. Anomaly based detection is currently mainly implemented as a support for zero-day network perimeter defence of big infrastructures and network operators, while signature based intrusion prevention remains the main mode of defence for most businesses and households. Pattern recognition or anomaly detection can be seen as classification problems. Classification problems refer to the problems in which the variable to be predicted is categorical. In network traffic the benign data is most often represented by a large number of examples, while malignant traffic appears extremely rarely or is an absolute rarity. This is known as the class imbalance problem and is a known obstacle to the induction of good classifiers by Machine Learning (ML) algorithms (Batista *et al.*, 2004).

He and Ma (2013) define imbalanced learning as the learning process for data representation and information extraction with severe data distribution skews to develop effective decision boundaries to support the decision-making process. He and Ma (2013) introduced informal conventions for imbalanced dataset classification. A dataset where the most common class is less than twice as common as the rarest class would be marginally unbalanced. A dataset with the imbalance ratio of about $10:1$ would be modestly imbalanced, and a dataset with imbalance ratios above $1000:1$ would be extremely unbalanced. This sort of imbalance is found in medical record databases regarding rare diseases, or production of electronic equipment, where non-faulty examples heavily outnumber faulty examples. Cases when negative to positive ratios are close to or higher than $1\,000\,000:1$ are called absolute rarity imbalance. This sort of imbalance is found in cyber security, where all but a few network traffic flows are benign. However, standard ML algorithms are still capable of inducing good classifiers for extremely imbalanced training sets. This shows that class imbalance is not the only problem responsible for the decrease in performance of learning algorithms. Batista *et al.* (2004) have demonstrated that a part of the problem to have class separation is often an overlap of classes due to a lack of feature separation. Another reason could be a lack of attributes, specific to a certain decision boundary. It is known that in cases where negative class has an internal structure (multimodal class), an overlap between negative and positive classes can be observed on a few of the clusters within negative class.

This study reports results of the empirical research executed with selected supervised machine learning classification algorithms in an attempt to compare their efficiency for intrusion detection and get improved results compared to other published studies. The study consists of the following sections: Section 2, introduction of the data sources, Section 3, a review of machine learning methods and model benchmark metrics used in this study, Section 4, an overview of the experiment and pre-processing steps, Section 5, results and conclusions.

1.1. *Contribution*

The research question raised in this study is *which supervised machine learning method consistently provides the best multi-class classification results with large and highly imbalanced network datasets*. To answer this question we chose the CIC-IDS2017, CSE-CIC-IDS2018 (Sharafaldin *et al.*, 2018) and LITNET-2020 (Damasevicius *et al.*, 2020) datasets as they are recent realistic software-generated traffic network datasets and meet the required criteria (Gharib *et al.*, 2016) for a good network intrusion dataset. An answer to this question is that based on rankings of performance metrics and bias-variance decomposition the tree ensembles *Adaboost, RandomForest Trees and Gradient Boosting Classifier* performed best on the network intrusion datasets used in this research.

The novelty of this research is in a proposed methodology (see Section 4) and application of it for the recent and not yet in depth studied dataset LITNET-2020. A review of the LITNET-2020 dataset compliance to the criteria raised by Gharib *et al.* (2016) is first introduced in Section 2.2. A variant of random under-sampling (skewed ratio under-sampling, proposed by authors and discussed in Section 3.1) is used to reduce imbalance of classes in a nonlinear fashion. SMOTE up-sampling for numeric data and SMOTE-NC for categorical data (see Section 3.2) is executed to increase representation of rare classes. Further in this research, comparison of multi-class classification performance of the CIC-IDS2017 and CIC-IDS2018 datasets with the LITNET-2020 dataset is discussed in Section 5. Multi-class performance *macro-averaged* metrics are implemented in this research. Balanced accuracy (Formula (2)) and geometric mean of recall (Formula (4)) for the LITNET-2020 dataset are implemented for the first time (see results in Tables 16 and 17). Multi-criteria scoring is cross-validated with an approach of testing through data previously unseen for the models (see Section 4). For decision tree ensemble methods, instead of the weak *CART* base classifiers, parameters *Tree depth* and *alpha* were Gird-Searched and validated using the method of maximum cost path analysis (Breiman *et al.*, 1984), see Section 3.8. Additional ML model, Gradient Boosting Classifier, utilizing ensemble of Classification and regression trees (CART), was introduced for benchmark in this research via the use of *XGBoost* library (Chen and Guestrin, 2016) with GPU support (see Section 3.5.6). In our methodology, due to the highly imbalanced nature of the used data, cost sensitive method implementations were chosen. These choices lead to better results (see Table 20) compared to other reviewed studies. Furthermore, selection of models with better generalization capabilities in this research is achieved through decomposition of classification error into bias and variance (see results in Table 18).

2. Datasets Used

The following section presents a review of datasets considered for this research together with arguments for the choice made.

2.1. *Datasets Considered for Analysis*

There are many datasets that have been used by the researchers to evaluate the performance of their proposed intrusion detection and intrusion prevention approaches. Far from being complete, the list includes: DARPA 1998 (Lippmann *et al.*, 1999) and 1999 traces by Lincoln Laboratory, USA, KDD'99 (Hettich and Bay, 1999), CAIDA (The Cooperative Association for Internet Data Analysis, 2010) datasets by University of California, USA, the Internet Traffic Archive and LBNL traces by Lawrence Berkeley National Laboratory, USA (Lawrence Berkeley National Laboratory, 2010), DEFCON by The Shmoo Group (2011), ISCX IDS 2012 (Shiravi *et al.*, 2012), CIDDS-001 (Coburg Intrusion Detection Data Set) (Ring *et al.*, 2017) and others. However, it has been widely acknowledged that machine learning research in an intrusion detection area needs to include new attack types and therefore researchers should consider more recent data sources.

In this research, three recent network data sets, compliant to the criteria described further (see Section 2.2) suggested by their authors for intrusion detection research, are explored. The datasets chosen are CIC-IDS2017, CSE-CIC-IDS2018 (Sharafaldin *et al.*, 2018) by the University of Brunswick, Canada, and LITNET-2020 (Damasevicius *et al.*, 2020). These datasets are of significant volume, contain anonymized real academic network traffic and are suited for multiple purposes of machine learning. LITNET-2020 is a new dataset that is given particular attention in this research, with discussion of compliance to the dataset suitability as devised by Gharib *et al.* (2016).

2.2. *Requirements for Cybersecurity Datasets*

Criteria for building such datasets are discussed by Małowidzki *et al.* (2015), Buczak and Guven (2016), Maciá-Fernández *et al.* (2018), Ring *et al.* (2019), Damasevicius *et al.* (2020), and others.

Małowidzki *et al.* (2015) define the following features of a good dataset: it must contain recent data, be realistic, contain all typical attacks met in the wild, be labelled, be correct regarding operating cycles in enterprises (working hours), should be flow-based. Ring *et al.* (2019) contend that a good dataset should be comparable with real traffic and therefore have more normal than malicious traffic, since most of the traffic within a company is normal and only a small part is malicious. Detailed framework and analysis of criteria for such datasets is proposed by Canadian Institute for Cybersecurity (CIC) at the University of New Brunswik. Gharib *et al.* (2016) have proposed the eleven dataset selection criteria. These criteria are presented in Table 1. Following this publication of the criteria, CIC created a list of new datasets,[1] addressing issues of compliance to these criteria. Creation of the CSE-CIC-IDS2018 followed with improvements, such as decreasing number of duplicates and uncertainties. Thakkar and Lohiya (2020) in Sections 4.1 and 4.2, Tables 4 and 5, and Karatas *et al.* (2020) in Sections III.C (CIC-IDS21017) and III. D (CSE-CIC-IDS2018) provide discussion and support to these claims.

[1] See https://www.unb.ca/cic/datasets/index.html.

Table 1
Dataset compliance criteria by Gharib *et al.* (2016).

No.	Criteria
1.	Complete network configuration
2.	Complete traffic
3.	Labelled dataset
4.	Complete interaction
5.	Complete record
6.	Available protocols
7.	Attack diversity
8.	Anonymity
9.	Heterogeneity
10.	Feature set
11.	Metadata

2.3. *LITNET-2020 Compliance*

The LITNET-2020 dataset was selected for the current study as complying to most of the above mentioned requirements with some reservations regarding interaction completeness, heterogeneity and feature set completeness criteria.

These eleven criteria as applied to LITNET-2020 are discussed below.

1. Complete network configuration: In order to investigate the real course of attacks, it is necessary to test the real network configuration. All of the network flows in this dataset are received or generated at the Network of Lithuanian academic institutions LITNET.
2. Complete traffic: The dataset accumulates full packet flows from the source to the destination, which can be a workstation computer, router or another specialized service device.
3. Labelled dataset: The dataset is labelled into a single benign and 12 malignant classes. The benign class is not separately labelled into sub-classes, however, it could be done because the number of benign records is exceeding 36 million records and is close to 92% of the whole dataset.
4. Complete interaction: The correct interpretation of the data requires data from the entire network interoperability process. LITNET-2020 dataset, however, is a pure network traffic dataset with no correlated host memory or host log information.
5. Record completeness: The LITNET-2020 dataset is compliant with this requirement.
6. Various protocols: Records of 13 types of protocols for normal and 3 types of protocols for malignant traffic are available in the LITNET-2020 dataset.
7. Diversity and novelty of attacks: The dataset includes attack flows that were detected from 2019-03-06 first flow and 2020-01-31 last flow.
8. Anonymity: It is important that the simulated set contain data for which privacy is not important. The LITNET-2020 data set contains no personally identifiable data.
9. Heterogeneity: Data from different sources, such as network streams, operating system logs, or network equipment logs, memory images, must be available. LITNET-2020 is not compliant with this requirement.

Table 2
Dataset content split.

Record Type	CIC-IDS2017	CSE-CIC-IDS2018	LITNET-2020
Benign	80.3%	83.1%	92.0%
Malignant	19.7%	16.9%	8.0%

Table 3
Dataset imbalance.

Imbalance category[1]	CIC-IDS2017	CSE-CIC-IDS2018	LITNET-2020
Modest $<(10:1)$	8.16%	0.00%	0.00%
High $<(1000:1)$	11.39%	16.85%	7.83%
Extreme $>(1000:1)$	0.15%	0.08%	0.20%
Total Malignant	19.7%	16.9%	8.0%

[1]Share of records in imbalance category.

10. Feature Set/Attribute Linkage: It is important for the research that data from different types of sources for the same event be linked, for example, device memory view, network traffic, and device logs. LITNET-2020 is not compliant with this requirement as it contains no linked host sources.
11. Metadata and documentation: Information about attributes, how the traffic was generated or collected, network configuration, attackers and victims, machine operating system versions and attack scenarios are required to do the research. LITNET-2020 is documented in Damasevicius *et al.* (2020).

2.4. *Cybersecurity Dataset Imbalance Problem*

In datasets selected for the research, the benign class takes from 80% up to 92% of total records (see Table 2), and some small classes only have less than 0.001% (see Table 4). The following Table 2 is a summary of the data set imbalance of benign versus malignant records:

The following Table 3 presents the split of malignant classes and is a summary of dataset imbalance shares in accordance with the taxonomy described by He and Ma (2013):

The following Table 4 represents a summary of extremely imbalanced $(>1000:1)$ classes in the three selected datasets.

Various imbalance measures are discussed by Ortigosa-Hernández *et al.* (2017) in a study, dedicated to such measures. In Karatas *et al.* (2020), section III.E, authors review most practical to use imbalance ratios of several IDS datasets, including the CIC-IDS2017 and CSE-CIC-IDS2018.

Referring to Ortigosa-Hernández *et al.* (2017) and Karatas *et al.* (2020), the following Formula (1) can be used for the calculation of the imbalance ratio:

$$\text{Imbalance Ratio} = \rho = \frac{\max\{C_i\}}{\min\{C_i\}}, \tag{1}$$

where: C_i shows the data size in the class i.

Table 4
Extremely rare classes in the datassets.

CIC-IDS2017		CIC-IDS2018		LITNET-2020	
Class	Share	Class	Share	Class	Share
Bot	0.0695%	DoS-Slowloris	0.0677%	W32.Blaster	0.0660%
Brute Force-Web	0.0532%	LOIC-UDP[1]	0.0107%	ICMP Flood	0.0638%
Brute Force-XSS	0.0230%	Brute Force-Web	0.0038%	HTTP Flood	0.0630%
Infiltration	0.0013%	Brute Force-XSS	0.0014%	Scan	0.0170%
SQL Injection	0.0007%	SQL Injection	0.0005%	Reaper Worm	0.0032%
Heartbleed	0.0004%			Spam	0.0021%
				Fragmentation	0.0013%
Total Extreme $>(1\,000:1)$	0.15%		0.08%		0.20%

[1]DDOS attack.

For example, historical NSL-KDD has an imbalance ratio of 648, CIC-IDS2017 has an imbalance ratio of 112 287 and CSE-CIC-IDS2018 has a slightly better imbalance ratio of 53 887. LITNET-2020 has an imbalance ratio of 70 769.

While *imbalance ratios* are an important part of the discussion, the *absolute rarity* is another concept introduced by He and Ma (2013) for the case when there is not enough records to learn the class. If there is not enough information within the feature-scape, determination of decision boundary cannot be made. There are no such classes in the LITNET-2020 datasets, and the data was sufficient for learning to all the machine learning algorithms used in our experiment. However, *Infiltration*, *Heartbleed* and *Web Attack-Aql Injection* classes in the CIC-IDS2017 dataset exhibit behaviour of such an absolute rarity and learning the decision boundaries for these classes is complicated and unspecific. In CSE-CIC-IDS2018 dataset, even though *Infiltration* class records are abundant, high overlap with benign class is observed.

2.5. *CIC-IDS-2017*

The CIC-IDS-2017 dataset (Sharafaldin *et al.*, 2018) is made available by Canadian Institute for Cyber Security Research at the University of New Brunswick[2] and introduces labelled data of 14 types of attacks including *DDoS*, *Brute Force*, *XSS*, *SQL Injection*, *Infiltration*, and *Botnet*. The traffic was emulated in a test environment during a period from July 3 to July 7, 2017. Network traffic features and related aggregates were extracted and generated using the CICFlowMeter tool and made available in a form of 8 CSV files. The CICFlowMeter is an open source tool[3] provided by CIC at UNB that generates bidirectional flows from pcap files, and extracts features from these flows, made available to the research community by Draper-Gil *et al.* (2016) and further described by Lashkari *et al.* (2017). The dataset contains a total of 2 830 743 records with flow data, synthetic features and is labelled.

The following Table 5 is a summary of class representation of this dataset.

[2]More information at https://www.unb.ca/cic/datasets/ids-2017.html.
[3]More information at https://www.unb.ca/cic/research/applications.html.

Table 5
Class representation in CIC-IDS2017 dataset.

Traffic class	Record count	Share (%)
BENIGN	2 273 097	80.3004%
DoS Hulk	231 073	8.1630%
PortScan	158 930	5.6144%
DDoS	128 027	4.5227%
DoS GoldenEye	10 293	0.3636%
FTP-Patator	7 938	0.2804%
SSH-Patator	5 897	0.2083%
DoS slowloris	5 796	0.2048%
DoS Slowhttptest	5 499	0.1943%
Bot	1 966	0.0695%
Web Attack-Brute Force	1 507	0.0532%
Web Attack-XSS	652	0.0230%
Infiltration	36	0.0013%
Web Attack-SQL Injection	21	0.0007%
Heartbleed	11	0.0004%

Dataset features, all measures of duration or related aggregates, further used for this research belong to these categories:

- Fiat (Forward Inter Arrival Time mean, min, max, std): aggregates on the time between two flows are sent in forward direction;
- Biat (Backward Inter Arrival Time mean, min, max, std): aggregates on the time between two flows are sent backwards;
- Flowiat (Flow Inter Arrival Time, mean, min, max, std): aggregates on the time between two flows sent in either direction;
- Active (mean, min, max, std): aggregates on the amount of time a flow was active before going idle;
- Idle (mean, min, max, std): aggregates on the amount of time a flow was idle before becoming active;
- Flow Bytes/s: Flow bytes sent per second;
- Flow Packets/s: Flow packets sent per second;
- Duration: The duration of a flow.

2.6. *CSE-CIC-IDS2018*

The CSE-CIC-IDS2018 dataset (Sharafaldin *et al.*, 2018) is made available by Canadian Institute for Cyber Security Research at the University of New Brunswick.[4] Data was emulated in the CIC test environment within an environment of 50 attacking machines, 420 victim PC's and 30 victim servers during the period from February 14 to March 2, 2018. The dataset contains records from 14 distinct attacks, is labelled and presented together with anonymised PCAP[5] files. 80 network traffic features were extracted and calculated

[4] More information at https://www.unb.ca/cic/datasets/ids-2018.html.

[5] File format as abbreviated from Packet CAPture, traffic capture file format in use by networking tools.

Table 6
Class representation of CSE-CIC-IDS2018 dataset.

Traffic class	Record count	Share (%)
Benign	13 484 708	83.070%
HOIC[1]	686 012	4.226%
LOIC-HTTP[1]	576 191	3.550%
Hulk[1]	461 912	2.846%
Bot	286 191	1.76%
FTP-BruteForce	193 360	1.191%
SSH-Bruteforce	187 589	1.156%
Infilteration	161 934	0.998%
SlowHTTPTest[1]	139 890	0.862%
GoldenEye[1]	41 508	0.256%
Slowloris[1]	10 990	0.068%
LOIC-UDP[1]	1 730	0.011%
Brute Force-Web	611	0.004%
Brute Force-XSS	230	0.001%
SQL Injection	87	0.0005%

[1] Variants of DoS attacks.

using the CICFlowMeter tool. Ten CSV files are made available for machine learning, containing 16 232 943 records. The representation of classes in IDS-2018 ranges from approximately 1 : 20 to 1 : 100 000.

The following Table 6 presents a summary of class representation of this dataset.

Same dataset features as described in Section 2.5 are used further in this research for selection of features.

2.7. *LITNET-2020*

LITNET-2020 is a new annotated network dataset for network intrusion detection, obtained from the real life Lithuanian academic network LITNET traffic by researchers from Kaunas Technology University (KTU). The environment of data collection, comparison of the dataset with other recently published network-intrusion datasets and description of attacks represented in the LITNET-2020 dataset is introduced by Damasevicius *et al.* (2020). The dataset contains benign traffic of the academic network and 12 attack types generated at KTU managed LITNET network from March 6, 2019 to January 31, 2020. Network traffic was captured using the open source nfcapd binary format, anonymised and processed into the CSV format, containing 39 603 674 time-stamped records. Nfsen, MeSequel, and Python script tools were used for extra feature generation and preprocessing, with data fields in CSV format named after fields, generated by Nfdump.[6] The 49 attributes that are specific to the NetFlow v9 protocol as defined in RFC 3954 (Claise, 2004) are used to form a dataset basis, further expanded with additional fields of time and tcp flags (in symbolic format), which can be used to identify attacks. An additional

[6] For a definition of features used in Nfdump 1.6 see https://github.com/phaag/nfdump/blob/master/bin/parse_csv.pl.

Table 7
Class representation of LITNET-2020 dataset.

Traffic class	Record label	Record count[1]	Share, %
Benign	none	36 423 860	91.9709%
SYN Flood	tcp_syn_f	1 580 016	3.9896%
Code Red	tcp_red_w	1 255 702	3.1707%
Smurf	icmp_smf	118 958	0.3004%
UDP Flood	udp_f	93 583	0.2363%
LAND DoS	tcp_land	52 417	0.1324%
W32.Blaster	tcp_w32_w	24 291	0.0613%
ICMP Flood	icmp_f	23 256	0.0587%
HTTP Flood	http_f	22 959	0.0580%
Port Scan	tcp_udp_win_p	6 232	0.0157%
Reaper Worm	udp_reaper_w	1 176	0.0030%
Spam botnet	smtp_b	747	0.0019%
Fragmentation	udp_0	477	0.0012%

[1]Record counts before removing timestamp and related record duplicates.

19 attack specific attributes are added. The representation of classes in LITNET-2020 is imbalanced in a range from approximately 1 : 30 to 1 : 100 000.

The following Table 7 presents a summary of class representation of this dataset.

3. Methods

Multiple different types of methods were used in this research to improve performance of ML methods. The methods employed could be grouped into pre-processing (see Sections 3.1–3.3) and machine learning methods (see Section 3.5). Data record sampling methods are discussed in detail in Section 3.1. Record over-sampling – in Section 3.2, feature selection, scaling and frequency transformation undertaken and pre-processing activities are discussed in Section 3.3. Machine learning methods (see Section 3.5), capable of cost sensitive learning, were chosen for performance comparison in this paper.

For all models, their hyper-parameters were searched using the *GridSearch* method, and later multiple performance measures (see Section 3.6) were used to evaluate and compare ML algorithms.

3.1. *Under-Sampling Methods*

The benign class in our datasets constitutes up to 90% of total records. Fixed ratio random under-sampling, utilizing uniform distribution for record selection, of benign and over-represented malignant class records was implemented on data load for all datasets. Under-sampling refers to the process of reducing the number of samples in a dataset. Fixed ratio random under-sampling method aims to balance class distribution through the random-uniform elimination of majority class examples. It is worth noting that random under-sampling can discard potentially useful data that could be important for the machine learning process. Under-sampling methods can be categorized into two groups: (i) fixed

ratio under-sampling and (ii) cleaning under-sampling (Lemaitre *et al.*, 2016). Fixed ratio under-sampling is based on a statistically random selection, which targets the provided absolute record numbers of a given class or a ratio, constituting a proportion of the total number of labels. Cleaning under-sampling is based on either (i) clustering, (ii) the nearest neighbour analysis, or (iii) classification accuracy (based on instance hardness threshold, Smith *et al.*, 2014).

Cleaning under-sampling approaches do not target a specific ratio, but rather clean the feature space based on some empirical criteria (Lemaitre *et al.*, 2016). According to Lemaitre *et al.* (2016), these criteria are derived from the nearest neighbour rule, namely: (i) condensed nearest neighbours (Hart, 1968), (ii) edited nearest neighbours (Wilson, 1972), (iii) one-sided selection (Kubat and Matwin, 1997), (iv) neighbourhood cleaning rule (Laurikkala, 2001), and (v) Tomek links (Tomek, 1976).

Cleaning under-sampling methods such as *Edited Nearest Neighbours*, *TomekLinks*, *Condensed Nearest Neighbours* were tested, however, due to the size of sub-sampled data and the large computational overhead they require, these methods were not further explored. The fixed random under-sampling was implemented in two steps as follows:

1. Major class records were first randomly under-sampled to a target number of records, such as to provide sufficient learning for all models. Target numbers were obtained after analysis of learning curves. Sufficient learning is defined here as the objective to have learning and testing curves to converge within a margin less than 1%, which for all models in this experiment occurs after approximately 0.6 million records.
2. Numbers of benign and other highly imbalanced classes were further transformed with a *random under-sampling* function from *Imbalanced-learn* library (Lemaitre *et al.*, 2016) using the number of records per class targets, calculated with the following empirically chosen skewed ratio function $N * (1 - \sqrt(s)/2)$ introduced in this research, where N is a number of initial records within a named class, where s is a share of records in that class. This proposed under-sampling method further on in this paper is referred to as *Skewed fixed ratio under-sampling*. The effect of this function is such that numbers of over-represented classes are decreased in a non linear manner, penalizing the best represented classes, while leaving the rare classes almost intact, thus simplifying, speeding up and decreasing the imbalance of the related learning of rare classes.

3.2. *Over-Sampling Methods*

In this paper, to balance minority classes, we investigate random and SMOTE (*Synthetic Minority Over-sampling Technique*) (Chawla *et al.*, 2002) over-sampling methods. Random over-sampling is a base method that aims to balance class distribution through the random replication of minority class examples. Unfortunately, this can increase the likelihood of classifier overfitting (Batista *et al.*, 2004). Therefore, we removed all duplicates in training data.

A more advanced method, capable of increasing minority class size without duplication, is SMOTE. SMOTE forms new minority class examples by linearly interpolating between minority class examples that are close. Thus, the overfitting problem risk is mitigated as the decision boundaries of the classifier for the minority class are moved further

away from the minority class space. SMOTE works in feature space, not in data space, therefore, before the procedure to over-sample is executed, the first step is to select numeric features to over-sample, as it is not necessary to over-sample in all dimensions. SMOTE over-sampling is achieved by following these steps: a) take k nearest neighbours from minority class for some minority class vector in the feature space, b) randomly choose the vector from those k neighbours, c) take a difference between the vector and its neighbour, and multiply the difference vector by a random number which lies between 0, and 1, d) repeat previous step until the target number of synthetic points is reached. After this, new records can be added to the current data (see Chawla *et al.*, 2002, for a complete algorithm). SMOTE method can be combined with some under-sampling methods to remove examples of all classes that tend to be misclassified. For example, in SMOTE with the *Edited Nearest Neighbours* (*ENN*) algorithm (Batista *et al.*, 2004), after SMOTE is used to over-sample a number of records in defined minority classes, *ENN* is used to remove samples from both classes such that any sample that is misclassified by its given number of nearest neighbours is removed from the training set. Batista *et al.* (2004) have demonstrated the best results on imbalanced datasets with minority classes containing under 100 records. However, due to the complexity of the edited neighbours procedures (Witten *et al.*, 2005) being $\mathcal{O}(nkl)$, where n is a number of samples, d – a number of dimensions (features) and k – a number of nearest neighbours, this solution is resource intensive.

As our datasets have not only continuous but also nominal features, we used a modification of SMOTE – *Synthetic Minority Over-sampling Technique-Nominal Continuous (SMOTE-NC)*, from *imbalanced-learn* library (Lemaître *et al.*, 2017) in the research. We used a recommended number of neighbours equal to $k = 5$, and separated categorical and numeric features before over-sampling.

3.3. *Feature Selection Methods*

Based on the ideas of research and practical implementation recommendations made by Sharafaldin *et al.* (2018) and Shetye (2019), a selection of features was tested with 3 classes of methods: (a) filtering – correlation and related heat map analysis (b) univariate – recursive feature elimination and (c) iterative – regularization methods. In this research, features were selected with *SelectKBest* from *Scikit-learn* library (Pedregosa *et al.*, 2011). The SelectKBest method takes as a parameter a score function, such as χ^2 or Anova *F-value*, or information gain function and retains the first k features with the highest scores.

If the Anova *F-value* function is used, a test result is considered statistically significant if it is unlikely to have occurred by chance, assuming the truth of the null hypothesis. If χ^2 is used as a score function, *SelectKBest* will compute the χ^2 statistic between each feature of X and y (assumed to be class labels). A small value will mean the feature is independent of y. A large value will mean the feature is non-randomly related to y, and so likely to provide important information. Only k features will be retained. Mutual information (information gain) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, whereas higher values mean higher dependency. Mutual information methods can capture any kind of statistical dependency, but being non-parametric

(Ross, 2014), it requires more samples for accurate estimation and is computationally more expensive, therefore, as a result of a better time performance in this research, Anova *F-value* was selected.

Embedded methods penalize a feature based on a coefficient threshold. On each iteration of the model training process those features which contribute the most to the training for a particular iteration are selected.

Further on in this paper, two methods, the filtering and *SelectKBest* from *Scikit-Learn* were used to select features.

When performing feature selection, *SelectKBest* is focusing on the largest classes, therefore a possible improvement would be to do feature selection in a pipeline, by firstly selecting the most important features for the rarest class and then adding features needed for every class.

Generating additional synthetic features was not attempted in this research, as all chosen datasets contain a significant number of such.

3.4. *Cost-Sensitive Learning Methods*

Cost-sensitive learning is a subfield of machine learning that takes the costs of prediction errors (and potentially other costs) into account when training a machine learning model (Brownlee, 2020).

If not configured, machine learning algorithms assume that all misclassification errors made by a model are equal. In case of an intrusion detection problem, missing a positive or minority class case is worse than incorrectly classifying an example from the negative or majority class.

The simplest and most popular approach to implementing cost-sensitive learning is to penalize the model less for training errors made on examples from the minority class by adjusting weights. The decision tree algorithm can be modified to weight model error by class weight when selecting splits. The Heuristic rule, also confirmed with intuition from decision trees (Brownlee, 2020), is to invert the ratio of the class distribution in the training dataset.

In this research, weights adjustment for decision trees was implemented using *Scikit-learn* library model parameters *class_weight*, setting it to *'balanced'*, which does the above mentioned inversion of class weights. Prior statistics were used for *Quadratic discriminant analysis* model.

3.5. *Choice of Machine Learning Methods*

For a performance comparison of machine learning methods on network intrusion detection data with imbalanced classes, we selected the most popular machine learning algorithms from surveys and review papers, related to intrusion detection (Buczak and Guven, 2016; Sharafaldin *et al.*, 2018; Damasevicius *et al.*, 2020).

3.5.1. *Adaptive Boosting (Adaboost)*

AdaBoost ensemble method was proposed by Yoav Freund and Robert Shapire for generating a strong classifier from a set of weak classifiers (Freund and Schapire, 1997). *AdaBoost*

algorithm works by weighting instances in the dataset by how easy or difficult they are to classify, and correspondingly prioritizes them in the construction of subsequent models. A Default base classifier was used with *Adaboost* by authors of the CIC-IDS-2017 dataset (Sharafaldin *et al.*, 2018) obtaining the result on *Precision* and F_1 of 0.77 whereas *Recall* at 0.84. Yulianto *et al.* (2019) used SMOTE, *Principal Component Analysis (PCA)*, and *Ensemble Feature Selection (EFS)* to improve the performance of *AdaBoost* on the CIC-IDS-2017 dataset achieving *Accuracy, Precision, Recall*, and F_1 scores of 0.818, 0.818, 1.000, and 0.900, respectively.

3.5.2. *Classification and Regression Tree (CART)*

The *Classification and Regression Tree* method was proposed by Breiman *et al.* (1984), and used to construct tree structured rules from training data. Tree split points are chosen on a basis of cost function minimization.

The authors of the CIC-IDS-2017 dataset (Sharafaldin *et al.*, 2018) obtained weighted averages of *Precision*, *Recall* and F_1 of 0.98 using ID3 (*Iterative Dichotomiser 3*), introduced by Quinlan (1986).

In this research, CART, as implemented in *Scikit-learn* library, was also used to obtain a base classifier and tree parameters for *Adaboost*, *Gradient Boosting Classifier* and *Random Forest Classifier*. Tree depth and alpha were obtained using the method of maximum cost path analysis (Breiman *et al.*, 1984), implemented in the *Scikit-learn* library cost-complexity-pruning-path function, discussed in Section 3.8.

3.5.3. *k-Nearest Neighbours (KNN)*

The *k-Nearest Neighbours* method was proposed by Dudani (1976), as a method which makes use of a neighbour weighting function for the purpose of assigning a class to an unclassified sample. KNN was used by authors of the CIC-IDS-2017 dataset (Sharafaldin *et al.*, 2018) with obtained results for weighted averages of *Precision, Recall* and F_1 of 0.96. The *KNN* algorithm in Scikit-learn by default uses the Euclidean distance as a distance metric for the k-NN algorithm. However, this is not appropriate when the domain presents qualitative attributes or categorical features of a different domain. For those domains, the distance for qualitative attributes is usually calculated using the overlap function, in which the value 0 (if two examples have the same value for a given attribute) or the value 1 (if these values differ) are assigned. In this research we have used the *Manhattan* dimension with positive effect obtained in the experiments.

3.5.4. *Quadratic Discriminant Analysis (QDA)*

Quadratic discriminant analysis descends from discriminant analysis introduced by Fisher (1954). Bayesian estimation for QDA was first proposed by Geisser (1964). *Quadratic discriminant analysis* (QDA) models the likelihood of each class as a Gaussian distribution, then uses the posterior distributions to estimate the class for a given test point (Friedman, 2001). The method is sensitive to the knowledge of priors. QDA was used by authors of the CIC-IDS-2017 dataset (Sharafaldin *et al.*, 2018) with obtained result for *Precision*, *Recall* and F_1 of 0.97, 0.88 and 0.92.

3.5.5. *Random Forest Trees (RFT)*

The *Random Forest Trees (RFT)* classifier was proposed by Breiman (2001) as a combination of tree predictors minimizing overall generalization error of participating trees as the number of trees in the forest becomes larger. Random forests are an alternative to *Adaboost* by Freund and Schapire (1997) and are more robust with respect to noise. Random Forests is an extension of bagged decision trees where only a random subset of features are considered for each split.

The algorithm was used by the authors of the CIC-IDS-2017 dataset (Sharafaldin *et al.*, 2018), and also by Kurniabudi *et al.* (2020). Sharafaldin *et al.* (2018) obtained results for the weighted averages of *Precision, Recall* and F_1 of 0.98, 0.97, and 0.97. In a study by Kurniabudi *et al.* (2020) the *Random Forest* algorithm has *Accuracy, Precision and Recall* of respectively 0.998 using the 15–22 selected features. These metrics were estimated for the benign and attack class.

3.5.6. *Gradient Boosting Classifier (GBC)*

In order to extend the scope of the research, *Gradient Boosting Classifier (GBC)*, as proposed by Friedman (2001) and Friedman (2002), was added as a natural member of classifier ensemble methods. GBC is a stochastic gradient boosting algorithm, where decision trees are fitted on the negative gradient of the chosen loss function. The idea of gradient boosting is to fit the base-learner not to re-weighted observations, as in *AdaBoost*, but to the negative gradient vector of the loss function evaluated at the previous iteration. *XGBoost* library (Chen and Guestrin, 2016), incarnation with GPU support of GBC, was implemented in this research. The results of GBC of other authors are not known publicly.

3.5.7. *Multiple Layer Perceptron*

Multiple Layer Perceptron (MLP) has been proposed by Rosenblatt (1962) as an extension to a linear perceptron model (Rosenblatt, 1957). It is a supervised learning artificial neural network implementation, utilizing back-propagation for training, that can have multiple layers and a chosen, non necessarily linear, activation function.

MLP was used in the study of Sharafaldin *et al.* (2018) with obtained results for weighted averages of *Precision, Recall* and F_1 of 0.77, 0.83, and 0.76.

3.6. *Performance Measures*

Standard performance metrics for classifiers are presented in Section 3.6.1, and *Bias and Variance decomposition* metric (see Section 3.7) was used to evaluate ML algorithm tendencies to overfit or underfit.

3.6.1. *Confusion Matrix Based Metrics*

Accuracy, *Precision* in equation (5), *Recall* in equation (3) and F_1 in equation (6), are very sensitive to the representation of classes in the source datasets (Sokolova and Lapalme, 2009). Results change if proportions of class samples change (Tharwat, 2018). In their study Garcia *et al.* (2010) review most of the performance measures used for imbalanced classes, introducing a new measure called *Index of Balanced Accuracy* (*IBA*) currently

implemented and used in the classification report of *Imbalanced-learn* library (Lemaitre *et al.*, 2016) for calculating *Geometric mean* of recall $\bar{G}$, equation (4) introduced by Kubat and Matwin (1997). An experimental comparison of performance measures for classification is presented by Ferri *et al.* (2009). Mosley (2013) reviews multi-class data performance metrics such as *Recall*, $\bar{G}$, *Relative Classifier Information* (*RCI*) (Wei *et al.*, 2010), *Matthew's Correlation Coefficient* (*MCC*) (Matthews, 1975), *Confusion Entropy* (*CEN*) (Jurman *et al.*, 2012). It is important to note that Chicco and Jurman (2020) demonstrated that *MCC* and *CEN* cannot be reliably used in case of an imbalance of data classes and these will not be discussed in this paper. Mosley (2013) introduces a *per-class Balanced Accuracy* (also known as *Balanced accuracy score (BAS)*), see equation (2) which is based on recall and neglects the precision. However, *Precision* is very sensitive to attributions of records from other classes, which was clearly observed during this research. In the case of imbalance, it mainly indicates a false classification of major classes, therefore, it has been chosen to be studied in this research.

Further on in this research, the *Balanced accuracy score* and $\bar{G}$ along with *Precision* were chosen as classification quality quantification metrics for comparison because: (i) these metrics were previously used by other researchers to measure performance of learning in imbalanced multi-class problems, while datasets used in this studyx have extremely imbalanced class distributions, (ii) these measures are available in popular and open source software libraries like *Scikit-learn* and *Imbalanced-learn*, (iii) metrics have simple and clear intuition for use in practical cyber-security applications, (iv) precision also allows for comparison with other research. *Macro* score averages were calculated in further experiment to give equal weight to each class, avoiding of scaling with respect to number of instances per class.

Balanced accuracy score *BAS* in formula (2) is further defined as average of recall values for K classes:

$$BAS = \frac{1}{K}\sum_{i=1}^{K} Recall_i, \tag{2}$$

where:

$$Recall_i = \frac{TP_i}{TP_i + FN_i} = \frac{c_{ii}}{\sum_{j=1}^{k} c_{ij}}, \tag{3}$$

where *TP* stands for True Positive, and *FN* stands for False Negative, i is a number of class in question and k is the number of classes in the dataset. TP_i is True Positive (correct classified) for class i, and FN_i are all false negative instances for the class i. c_{ij} is an element of the confusion matrix in row i and column j.

Geometric mean $\bar{G}$ of sensitivity is defined as follows:

$$\bar{G} = \sqrt[k]{\prod_{i=1}^{k} \text{Recall}_i}, \tag{4}$$

where k is a number of classes in a dataset.

Precision for class i is defined as follows:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} = \frac{c_{ii}}{\sum_{j=1}^{k} c_{ji}}. \tag{5}$$

Whereas F_1 for class i is defined as follows:

$$F_{1i} = \frac{2}{\frac{1}{Precision_i} + \frac{1}{Recall_i}}. \tag{6}$$

In this research, we have used macro-weighted (i.e. unweighted mean) $\bar{G}$, *Precision* and F_1, if it is not specified otherwise.

3.7. *Bias and Variance Decomposition*

The decomposition of the loss into bias and variance helps to improve understanding of generalization capacities of compared learning algorithms, such as overfitting and underfitting. Various methods of decomposition are reviewed in Domingos (2000). It has been demonstrated that high variance correlates to overfitting, and high bias correlates to underfitting. In practical terms, when comparing the performance of learning algorithms, models with lower bias and variance over the same test data would be preferred. It is worth noting that models with a higher degree of parameter freedom tend to demonstrate lower bias and higher variance, and models with a low degree of freedom demonstrate high bias and lower variance.

The loss function of a learning algorithm can be decomposed into three terms: a variance, a bias, and a noise term, which will be ignored further for simplicity (Raschka, 2018). Loss function depends on the machine learning algorithm. For decision trees (*CART*), training proceeds through a greedy search, each step based on information gain. For the random forest classifier, loss function is the *Gini impurity*. *Cross-entropy* is the default loss function to use for multi-class classification problems with *MLP*.

The prediction bias is calculated as the difference between the expected prediction accuracy of a model and the true prediction accuracy (equation (7)). In formal notation the $Bias$ of an estimator $\hat{\beta}$ is the difference between its expected value $E[\hat{\beta}]$ and the true value of a parameter β being estimated (Raschka, 2018):

$$Bias = E[\hat{\beta}] - \beta. \tag{7}$$

The variance (equation (8)) is a measure of the variability of model's predictions if the learning process is repeated multiple times with random fluctuations in the training set.

$$Variance = E\left[\left(\hat{\beta} - E[\hat{\beta}]\right)^2\right]. \tag{8}$$

Variance is obtained by repeating prediction on a model trained on stratified shuffle-split training data. The more sensitive the model-building process is towards fluctuations of the training data, the higher the variance (Raschka, 2018).

3.8. *Tree Pruning*

Finding the values where training and testing learning curves converge allows for creation of better generalizing decision trees, decrease of overfitting and underfitting. The Tree depth (implemented in *Scikit-learn* library through parameter *max_depth*) and α (implemented in *Scikit-learn* library through parameter *ccp_alpha*) were obtained using the method of maximum cost path analysis (Breiman *et al.*, 1984), implemented in *Scikit-learn* library *cost-complexity-pruning-path* function and searching for a minimum of Bias and Variance. In this algorithm the cost-complexity measure $R_\alpha(T)$ of a given tree T is defined in formula (9) as follows:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|, \tag{9}$$

where $|\tilde{T}|$ is the number of terminal nodes in T, $R(T)$ is defined as the total misclassification cost of the terminal nodes for the complexity parameter α ($\geqslant 0$). As α increases, more descendent nodes are pruned.

3.9. *Variance Inflation Factor*

Many variables in the datasets CIC-IDS2017 and CSE-CIC-IDS2018 appear to be correlated with each other, which increases bias while using *Quadratic Discriminate Analysis*. A statistical measure known as *VIF* (Variance Inflation Factor) was proposed by Lin *et al.* (2011) to support elimination of cross-correlation of features and is implemented in this research from *statsmodels* library (Seabold and Perktold, 2010).

3.10. *Other Methods*

The number of estimators was obtained using the *Scikit-learn's GridSearch* (LaValle *et al.*, 2004) method. See Sections 4.4–4.5 and Table 15 for implementation details in this research.

4. Experiment Design

Our experiment contained pre-processing, described further in detail in Section 4.1 for the CIC-IDS2017 dataset, Section 4.2 for the CSE-CIC-IDS2018 dataset and Section 4.3 for the LITNET-2020 dataset. The datasets were cleaned and normalized. *Quantile* transformation from *Scikit-learn* library (Pedregosa *et al.*, 2011) with *QuantileTransformer* using a default of 1 000 quantiles has been implemented for the pre-processing of numeric (continuous time related) features of all datasets in order to transform original values to a more uniform distribution.

Datasets were further under-sampled with random fixed ratio under-sampling and proposed skewed fixed ratio under-sampling so that after splitting into testing and training, sets would contain more than approximately 600 000 records each, which is sufficient for

learning of all algorithms. This number has been estimated by performing learning curve analysis.

Later on, the training subsets were over-sampled using SMOTE for CIC-IDS2017 and CIC-IDS2018 datasets and SMOTE-NC for LITNET-2020. Features were selected using *KBest* (see Section 3.3) and *VIF* procedures (see Section 3.9). Training and hyper-parameter search was performed using cross validation with $CV = 20$ on stratified shuffle split samples of training datasets.

The final results of predictions were obtained using testing data, e.g. not seen to trained models. In order to obtain a reliable result, predictions were run 30 times with a change of random seed on each run.

Further on in the experiment, the best features were selected using the *SelectKBest* procedure from *Scikit-learn* library (Pedregosa *et al.*, 2011) and followed by Variance inflation factor analysis (Lin *et al.*, 2011) with a target threshold value, to eliminate variables with high collinearity.

Parameters for classification models were searched using *GridSearch* from the *Scikit-learn* library.

4.1. *CIC-IDS2017 Pre-Processing Steps*

The following procedures were implemented to condition the dataset for better learning of under-represented attack classes: a) removal of unused features and related record duplicates, b) random under-sampling of benign class records, such as to represent no more than a number of records, providing sufficient learning for the worst performing model, obtained after analysis of learning curves and c) over-sampling using SMOTE for the training sub-sample of extremely rare records (see Table 4) up until the minimum number of examples of classes with high imbalance.

Duplicate rows were removed (leaving the first one), see Table 8.

The following 8 features '*Bwd PSH Flags*', '*Bwd URG Flags*', '*Fwd Avg Bytes/Bulk*', '*Fwd Avg Packets/Bulk*', '*Fwd Avg Bulk Rate*', '*Bwd Avg Bytes/Bulk*', '*Bwd Avg Packets/Bulk*', '*Bwd Avg Bulk Rate*', containing no information ($Std = 0$) in all loaded files and duplicate feature '*Fwd Header Length.1*' ($corr = 1$) with '*Fwd Header Length*' were removed.

After dropping the duplicates, the 2 522 362 remaining records were investigated for missing values and infinities.

As a result, 1 358 missing values containing records were removed with drop duplicates. The remaining 353 rows with missing values were found to be split between '*Benign*' (350) and '*DoS Hulk*' (3) classes and missing values were replaced with -1.

Further, 1 211 records with infinities in two features Flow '*Bytes/s*' and '*Flow Packets/s*' were found and replaced by maximums of values per class, see Table 9.

This processing step is made under an assumption that such a replacement for lost values would be possible to implement after learning the values during the initial training of a real life intrusion detection system.

Further numbers of records for *Benign* and second largest class *Dos Hulk* were transformed with a skewed fixed ratio under-sampling. Remaining data is split into test and

Table 8
Removal of duplicates in IDS2017 dataset.

Class	Share of removed records (%),	Resulting counts[1]	Resulting share (%)
Benign	7.770%	2 096 484	83.1159%
DoS Hulk	25.197%	172 849	6.8527%
PortScan	42.856%	90 819	3.6006%
DDoS	0.009%	128 016	5.0752%
DoS GoldenEye	0.068%	10 286	0.4078%
FTP-Patator	25.258%	5 933	0.2352%
SSH-Patator	45.413%	3 219	0.1276%
DoS slowloris	7.091%	5 385	0.2135%
DoS Slowhttptest	4.928%	5 228	0.2073%
Bot	0.661%	1 953	0.0774%
Web Attack – Brute Force	2.455%	1 470	0.0583%
Web Attack-XSS	0.000%	652	0.0258%
Infiltration	0.000%	36	0.0014%
Web Attack-Sql Injection	0.000%	21	0.0008%
Heartbleed	0.000%	11	0.0004%
Total		2 522 362	

[1]Record counts after removing duplicate records.

Table 9
Replacing infinities in IDS2017 dataset.

Class	Record count	Flow Bytes/s	Flow Packets/s
Benign	1 077	2.071e+09	4.0e+06
PortScan	125	8.00e+06	2.0e+06
Bot	5	1.20e+07	2.0e+06
FTP-Patator	2	1.40e+07	3.0e+06
DDoS	2	3.47e+08	2.0e+06
Total:	1211		

train sub-samples. The training sub-set is then over-sampled with SMOTE (thus training record count values of 4 999 and 2 999 in Table 10). This procedure keeps all extremely imbalanced class records (Table 4) intact and adds new records for the training, resulting in record counts for the training and testing samples presented in Table 10.

After this, the values of numeric columns were scaled to a range of [0; 1] with *Scikit-learn* (Pedregosa *et al.*, 2011) *QuantileTransform*. This transformation assigns each feature into a quantile individually and scales such that it is in the given range on the training set, by default between zero and one.

Further in this research, the 40 best features were selected using the *SelectKBest* procedure from the *Scikit-learn* library (Pedregosa *et al.*, 2011) and followed by *Variance inflation factor analysis* with a target threshold value equal to 40, to eliminate variables with high collinearity.

4.2. *CIC-IDS2018 Pre-Processing Steps*

The same pre-processing procedure from Section 4.1 was applied to dataset CIC-IDS2018.

Table 10
Resulting IDS2017 dataset training and/or validation sample representation.

Record label	Training records	Resulting share (%)	Testing records	Resulting share (%)
Benign	442 421	64.739%	442 421	67.508%
DoS Hulk	86 425	12.646%	86 424	13.187%
DDoS	64 008	9.366%	64 008	9.767%
PortScan	45 410	6.645%	4 5409	6.929%
DoS GoldenEye	5 143	0.753%	5 143	0.785%
FTP-Patator	4 999	0.731%	2 967	0.453%
DoS slowloris	4 999	0.731%	2 692	0.411%
DoS Slowhttptest	4 999	0.731%	2 614	0.399%
SSH-Patator	4 999	0.731%	1 610	0.246%
Bot	4 999	0.731%	976	0.149%
Web Attack-Brute Force	2 999	0.439%	735	0.112%
Web Attack-XSS	2 999	0.439%	326	0.050%
Infiltration	2 999	0.439%	18	0.003%
Web Attack-Sql Injection	2 999	0.439%	11	0.002%
Heartbleed	2 999	0.439%	6	0.001%
Total:	683 397		655 360	

The timestamp column and related record duplicates were removed, as no time series dependent machine learning methods were chosen in this research.

Afterwards, 8 features '*Bwd URG Flags*', '*Bwd Pkts/b Avg*', '*Bwd PSH Flags*', '*Bwd Blk Rate Avg*', '*Fwd Byts/b Avg*', '*Fwd Pkts/b Avg*', '*Fwd Blk Rate Avg*', '*Bwd Byts/b Avg*' containing no information (eq. $Std = 0$) were removed.

The following sampling procedures were executed in order to achieve a better balance between major classes and extremely rare classes:

1. the top two classes ('*Benign*' and '*DDoS attacks-LOIC-HTTP*') were under-sampled so as to represent no more than a number of records, providing sufficient learning for the worst performing model, obtained after analysis of learning curves.
2. The remaining data was split into test and train sub-samples.
3. Training sub-set was then over-sampled with SMOTE (thus, value of 2 999). This procedure keeps all extremely imbalanced class records (Table 4) intact and adds new records for the training, resulting in record counts for the training and testing samples presented in Table 11.

It should be noted that 7 373 records with infinities in two features '*Flow Bytes/s*' and '*Flow Packets/s*' were found and replaced by maximums of values per class, see Table 12.

Presence of such values could indicate that related flows were not terminated on recording.

After the data cleaning, the dataset was normalized with *QuantileTransform*. The 40 best features from *SelectKBest* were passed through the *Variance Inflation Factor* procedure with a threshold of 40 which was selected to eliminate collinearity of features.

Table 11
Resulting IDS2018 dataset training and validation sample representation.

Record label	Training records	Resulting share (%)	Testing records	Resulting share (%)
Benign	134 850	20.067%	134 849	20.576%
DDoS attacks-LOIC-HTTP	129 558	19.280%	129 558	19.769%
DDOS attack-HOIC	99 430	14.796%	99 431	15.172%
Infilteration	72 612	10.805%	72 613	11.080%
DoS attacks-Hulk	72 599	10.804%	72 600	11.078%
Bot	72 268	10.754%	72 267	11.027%
SSH-Bruteforce	47 024	6.998%	47 024	7.175%
DoS attacks-GoldenEye	20 703	3.081%	20 703	3.159%
DoS attacks-Slowloris	4 954	0.737%	4 954	0.756%
DDOS attack-LOIC-UDP	2 999	0.446%	865	0.132%
Brute Force-Web	2 999	0.446%	285	0.043%
Brute Force-XSS	2 999	0.446%	114	0.017%
SQL Injection	2 999	0.446%	43	0.007%
FTP-BruteForce	2 999	0.446%	27	0.004%
DoS attacks-SlowHTTPTest	2 999	0.446%	27	0.004%
Total:	671 992		655 360	

Table 12
Replacing infinities in IDS2018 dataset.

Class	Record count	Flow Bytes/s	Flow Packets/s
Benign	6 243	1.47e+09	4.0e+6
Infilteration	1 129	2.74e+08	3.0e+06
FTP-BruteForce	1	0.0e+00	2.0e+06
Total:	7 373		

4.3. *LITNET-2020 Dataset Pre-Processing*

Due to the choice of supervised machine learning models and problem definition in this study, the LITNET-2020 dataset *timestamp* feature was not used. Features related to the source and destination address, such as source and destination issuing authorities, are highly supportive in discovering not only the attacker but also the attack class, therefore, in order to support generalization of training, they were eliminated.

After removing *timestamp* and address related features, related duplicate records were also removed, see Table 13.

The resulting dataset is even more imbalanced. The target number of records of the *Benign* and the *Code Red* type was set after learning curves that indicate the number of records required by the worst performing model for sufficient learning. Sufficient learning is defined here as the objective of getting the learning and testing curves to converge within a margin of less than 1%, which for all models under experiment occurs after approximately 0.5 million records.The dataset was further split by half into testing and validation.

As a final step, a Synthetic Minority Over-sampling Technique for Nominal and Continuous features for datasets with categorical features, SMOTE-NC, introduced by Chawla *et al.* (2002) was implemented, see Table 14.

Table 13
Removal of timestamp related duplicates in LITNET-2020 dataset.

Traffic type	Share of removed records (%)	Resulting counts of records[1]	Resulting share (%)
Benign	33.1%	24 349 750	95.052%
SYN Flood	98.2%	28 873	0.113%
Code Red	13.5%	1 085 656	4.238%
Smurf	87.7%	14 642	0.057%
UDP Flood	1.3%	92 412	0.361%
LAND DoS	75.3%	12 926	0.050%
W32.Blaster	99.2%	200	0.001%
ICMP Flood	92.6%	1 723	0.007%
HTTP Flood	1.7%	22 578	0.088%
Scan	0.0%	6 232	0.024%
Reaper Worm	0.3%	1 173	0.005%
Spam	0.1%	746	0.003%
Fragmentation	15.9%	401	0.002%

[1]Record counts after removing timestamp and related record duplicates.

Table 14
LITNET-2020 dataset sample representation.

Record label	Training records	Resulting share (%)	Testing records	Resulting share (%)
Benign	349 470	51.277%	349 470	53.325%
Code Red	215 484	31.618%	215 485	32.880%
UDP Flood	45 858	6.729%	45 859	6.997%
SYN Flood	14 436	2.118%	14 437	2.203%
HTTP Flood	11 289	1.656%	11 289	1.723%
Smurf	9 999	1.467%	7 321	1.117%
Scan	9 999	1.467%	6 463	0.986%
LAND DoS	9 999	1.467%	3 116	0.475%
Spam	2 999	0.440%	710	0.108%
Reaper Worm	2 999	0.440%	587	0.090%
ICMP Flood	2 999	0.440%	373	0.057%
Fragmentation	2 999	0.440%	153	0.023%
W32.Blaster	2 999	0.440%	100	0.015%
Total:	681 529		655 363	

After the data cleaning, the dataset was normalized with *QuantileTransform*. The 40 best features from *SelectKBest* were obtained and further checked for feature collinearity. Collinear features were reduced using the Variance Inflation Factor procedure (see Section 3.9) with a threshold value of 40.

4.4. *Experiment Software Environment*

All code for models was realized in the Python 3.7 environment on Anaconda 3 using *Scikit-learn*[7] and *Imbalanced-learn*[8] libraries, except for the *Gradient Boosting Classifier*,

[7]https://scikit-learn.org/stable/.
[8]https://imbalanced-learn.org/stable.

which was implemented using the *XGBoost* library (Chen and Guestrin, 2016), utilizing GPU.

Model parameters were searched with the *GridSearch* method. *Tree depth* and *alpha* were further validated using the method of maximum cost path analysis (Breiman *et al.*, 1984), implemented in *Scikit-learn* by the *cost-complexity-pruning-path* function (see Section 3.8).

4.5. *Parameter Values Selection*

The following parameter ranges were selected for the grid search:

1. ADA: *n_estimators: (range(10, 256, 5))*, *learning_rate: [0.001, 0.005, 0.01, 0.5, 1]*, and base estimator – CART.
2. CART: *criterion*: ('entropy', 'gini'), *max_depth: range(4, 32)*, *in_samples_leaf: range(6, 10, 1)*, *max_features: [0.5, 0.6, 0.8, 1.0, 'auto']*.
3. GBC: *max_depth: range(4, 32, 1)*, *n_estimators: range(100, 256, 5)*, other parameters used from *CART*.
4. KNN: *n_neighbors: range(3, 16, 1)*, *algorithm: ['ball_tree', 'auto']*, *leaf_size: range(15, 35, 5)*
5. MLP: *hidden_layer_sizes: tuple (32 ... 256, 32 ... 256)* (*step* = 1), *alpha: np.geomspace(1e−2, 2, 50, endpoint = True)*, *activation: ['identity', 'logistic', 'tanh', 'relu']*, *solver: ['lbfgs', 'sgd', 'adam']*, *learning_rate: ['constant', 'adaptive']*, *beta_1 : np.linspace(0.85, 0.95, 11, endpoint = True)*, *learning_rate_init: np.geomspace(2e−4, 6e−4, 5, endpoint = True)*, *max_iter: [200, 300]*, *early_stopping: [True, False]*.
6. QDA: *reg_param: np.geomspace(1e−19, 1e−1, 50, endpoint = True)*. Value of *tol* parameter only impacts threshold when warnings of variable collinearity should be suppressed.
7. RFC: *n_estimators: range(100, 350, 5)*, other parameters in the same ranges as *CART*.

The parameters used in this study are presented in the Table 15.

5. Results and Discussion

5.1. *Results of the Conducted Experiments*

Tables 16, 17 and 18 represent the results of ML methods rankings using a *Standard Ranking* approach (Adomavicius and Kwon, 2011), where equal items get the same ranking number, and a gap is left in between the smaller and bigger result, where the bigger result means a worse result.

In Table 16, the results of scoring by *Balanced Accuracy* are in favour of trees or their ensembles, *Adaboost* being the strongest, closely followed by *Random Forest Classifier* and *K-Nearest Neighbours*.

Results of this research support notion that *Balanced Accuracy* metric (see Table 16) should be used for measuring accuracy in case of highly and extremely imbalanced data

Table 15
Model parameters used.

Model	Dataset		
	CIC-IDS2017	CIC-IDS2018	LITNET-2020
	Parameters		
ADA	base_estimator = DecisionTreeClassifier, learning_rate = 1[1], n_estimators = 120, tree parameters as indicated for CART, next row		
CART	criterion = 'entropy', min_samples_leaf = 7, max_features = 0.5, max_depth = 32, ccp_alpha = 0.00001, class_weight = 'balanced'	criterion = 'entropy', min_samples_leaf = 7, max_features = 0.5, max_depth = 32, ccp_alpha = 0.00001, class_weight = 'balanced'	criterion = 'entropy', min_samples_leaf = 7, max_features = 0.5, max_depth = 15, ccp_alpha = 0.00001, class_weight = 'balanced'
GBC	n_estimators = 120, min_samples_leaf = 7, max_features = 0.5, max_depth = 15, ccp_alpha = 0.00001, tree_method = 'gpu_hist'	n_estimators = 120, min_samples_leaf = 7, max_features = 0.5, max_depth = 15, ccp_alpha = 0.00001, tree_method = 'gpu_hist'	n_estimators = 120, min_samples_leaf = 7, max_features = 0.5, max_depth = 15, ccp_alpha = 0.00001, tree_method = 'gpu_hist'
KNN	algorithm = 'ball_tree', leaf_size = 30[1], metric = 'manhattan' n_neighbors = 4, weights = 'distance'	algorithm = 'ball_tree', leaf_size = 30[1], metric = 'manhattan', n_neighbors = 4, weights = 'uniform'[1]	algorithm = 'ball_tree', leaf_size = 30[1], metric = 'minkowski'[1], n_neighbors = 4, p = 2[1], weights = 'uniform'[1]
MLP	activation = 'relu'[1], solver = 'adam'[1], alpha = 0.0[1], beta_1 = 0.9[1], hidden_layer_sizes = (120, 60), learning_rate = 'constant'[1], learning_rate_init = 0.001[1], early_stopping = True[1], max_iter = 200[1], warm_start = False[1]	activation = 'relu'[1], solver = 'adam'[1], alpha = 0.067, beta_1 = 0.86, hidden_layer_sizes = (32, 46), learning_rate = 'adaptive', learning_rate_init = 0.00045, early_stopping = False, max_iter = 300, warm_start = True	activation = 'relu'[1], solver = 'adam'[1], alpha = 0.0[1], beta_1 = 0.9[1], hidden_layer_sizes = (120, 60), learning_rate = 'adaptive', learning_rate_init = 0.001[1], early_stopping = True[1], max_iter = 200[1], warm_start = True
QDA	priors = priors[2], reg_param = 2.1e-8, tol = 0.1	priors = priors[2], reg_param = 2.3e-5, tol = 0.1	priors = priors[2], reg_param = 0.002, tol = 0.1
RFC	criterion = 'entropy', min_samples_leaf = 7, max_features = 0.5, max_depth = 15, n_estimators = 120, ccp_alpha = 0.0[1], class_weight = 'balanced'	criterion = 'entropy', min_samples_leaf = 7, max_features = 1.0, max_depth = 15, n_estimators = 120, ccp_alpha = 0.0[1], class_weight = 'balanced'	criterion = 'entropy', min_samples_leaf = 8, max_features = 0.5, max_depth = 15, n_estimators = 156, ccp_alpha = 0.00001, class_weight = 'balanced'

[1]Default *Scikit-Learn* values; [2]Priors calculated equal to class shares.

sets. *Error Rate* for all models is below 0.1, while *Balanced Accuracy* manifests some insufficient learning. *Accuracy* of *Extremely rare* (malicious) classes in this research is dominated by majority (benign) class, representing over 80% of the whole data (see Tables 2 and 3) and therefore *Error Rate* is overly optimistic, under-representing the prediction error of *Extremely rare classes* (see Table 4), important to this research.

Table 16
Comparison of Model performance on 3 datasets using Balanced Accuracy Score (BAS) and Error Rate (ErR).

	CIC-IDS2017			CIC-IDS2018			LITNET-2020			Rank by BAS	
Model	ErR	BAS	Rank	ErR	BAS	Rank	ErR	BAS	Rank	Total	Best
ADA[1]	0.001	0.995	1	0.060	0.887	4	0.003	0.996	1	5	1
CART	0.004	0.984	5	0.064	0.897	3	0.005	0.985	4	12	4
GBC	0.003	0.986	4	0.063	0.811	4	0.011	0.756	6	14	5
KNN	0.006	0.989	3	0.060	0.917	1	0.044	0.864	5	9	3
MLP	0.020	0.937	7	0.072	0.860	6	0.070	0.698	7	20	7
QDA	0.068	0.951	6	0.090	0.843	7	0.022	0.992	2	15	6
RFC	0.002	0.991	2	0.059	0.898	2	0.005	0.987	3	7	2

[1]Adaboost ensemble is made of CART estimators with the grid-searched hyper-parameters described in Table 15.

Table 17
Model rankings by Precision (Pr) and G-mean ($\bar{G}$).

	CIC-IDS2017			CIC-IDS2018			LITNET-2020			Rank	
Model	Pr	$\bar{G}$	Rank	Pr	$\bar{G}$	Rank	Pr	$\bar{G}$	Rank	Total	Best
ADA	0.928	0.919	1	0.991	0.990	1	0.970	0.994	1	3	1
CART	0.868	0.886	5	0.971	0.977	6	0.828	0.989	4	15	5
GBC	0.892	0.884	4	0.988	0.987	2	0.963	0.987	3	9	3
KNN	0.906	0.912	2	0.988	0.987	2	0.674	0.519	7	11	4
MLP	0.879	0.834	6	0.979	0.977	5	0.685	0.876	6	17	6
QDA	0.713	0.839	7	0.936	0.881	7	0.915	0.978	5	19	7
RFC	0.913	0.907	2	0.985	0.984	4	0.937	0.998	2	8	2

The ranking results in Table 17 were obtained based on the minimum of the sum of rankings for *Presicion* and $\bar{G}$. The results of scoring by *Precision and* $\bar{G}$ are in favour of the same tree ensembles.

The rankings of bias and variance decomposition in Table 18 are obtained on a basis of the minimum of the sum of bias and variance (equal to the model mean squared error, when not accounted for the noise component). The bias and variance are calculated according to formulas (7) and (8). To calculate bias, we have to estimate β and $\hat{\beta}$. β is equal to true class labels vector of test dataset. To estimate $\hat{\beta}$, the bootstrap with replacement of training dataset is taken 5 times, each time the model is trained and its prediction for each training dataset is stored as a separate vector $\hat{\beta}$ value. Then $Bias^2$ is estimated as squared length of the difference of average prediction vector ($E[\hat{\beta}]$) and test dataset true label vector (β) and divided by the number of test records. The variance (Var) is then calculated by formula (8), e.g. it estimates the variance in $\hat{\beta}$ calculated for each bootstrap sample with replacement from the training dataset.

The *QDA* values that are much higher than average compared to other algorithm errors from the same data in Table 18 are a characteristic property of models with low number of hyper-parameters as noted in Brownlee (2020). Values obtained in this experiment could be local optima, but authors were not able to find other parameter values that would result in lower difference of values for this model between datasets. However, bias and variance of this model was noticed to be sensitive to changes in a list of features selected before the

Table 18
Model rankings using model bias and variance (Var) decomposition.

	CIC-IDS2017			CIC-IDS2018			LITNET-2020			Rank[1]	
Model	Bias[2]	Var	Rank	Bias[2]	Var	Rank	Bias[2]	Var	Rank	Total	Best
ADA	0.09	0.024	1	1.36	0.324	1	0.22	0.006	1	3	1
CART	0.15	0.109	6	1.80	0.966	5	0.26	0.049	4	15	4
GBC	0.08	0.025	1	1.96	0.201	2	0.22	0.041	3	6	2
KNN	0.14	0.050	4	2.26	0.984	6	1.08	0.335	7	17	5
MLP	0.16	0.051	5	2.77	0.477	7	0.54	0.231	5	17	5
QDA	0.56	0.018	7	19.23	0.985	8	1.12	0.003	6	21	8
RFC	0.11	0.034	3	1.90	0.279	3	0.25	0.006	2	8	3

[1]Ranking is performed on the sum of model loss variance and bias squared; [2]Bias squared value.

parameter search process. The list of features chosen for model training is individual for each dataset.

5.2. *Discussion and Comparison of the Results*

Comparison of results of research in different implementations for CIC-IDS2017 and CSE-CIC-IDS2018 datasets is presented in Table 19. Performance metrics are not directly comparable to our research (further in Table 19 – this research), as validation results in our experiment were obtained using multiple class optimization and 50% of dataset as a hold-out data, versus standard k-fold cross-validation, known to be prone to knowledge leak. In our methodology, cost sensitive model implementations provided classification for multiple class measures. However, for comparison, traditional measures suitable only for balanced datasets are presented with other reviewed studies (see Table 19). It is important to note that optimization in this experiment was done on *Balanced Accuracy Score*, therefore, other measures are sub-optimal.

In Sharafaldin *et al.* (2018) authors had an objective to introduce the CIC-IDS-2017 dataset, and default parameter model results of machine learning are presented for purely benchmark purposes of future research. Feature selection was performed using the random forest regression feature selection algorithm. The results of *Precision*, *Recall* and F_1 were obtained in their studies in a form of *weighted average* of each evaluation metrics and are represented in Table 19. Iterative Dichotomiser 3, decision tree learner with an early stopping, as implemented in Weka (Witten and Frank, 2002), is used in their research. In our research the results were obtained using *macro average* for the above mentioned and other performed metrics. *Macro averages* of metrics are more sensitive to the imbalance of classes.

In Sharafaldin *et al.* (2019) authors improve results on RFT through proposing super-feature creation versus random feature regression algorithm for feature selection used in previous research (Sharafaldin *et al.*, 2018). In our research the feature selection was obtained through fast *Kbest* procedure with *Anova F-value* optimization function, however, algorithm has been chosen after testing three classes of feature selection methods.

In Yulianto *et al.* (2019) strategy, SMOTE is utilized with CIC-IDS-2017. However, only benign and DDos class data of CIC-IDS-2017 dataset is taken, calculating binary

Table 19
Related research results analysis.

Algorithm	Dataset	Precision	Recall	F1	Source[1]
ADA	CIC-IDS-2017	0.77	0.84	0.77	(Sharafaldin *et al.*, 2018)
ADA	CSE-CIC-IDS2018	0.999	0.999	0.999	(Kanimozhi and Jacob, 2019a)
ADA	CIC-IDS-2017	0.818	1.0	0.900	(Yulianto *et al.*, 2019)
ADA	CSE-CIC-IDS2018	0.997	0.997	0.997	(Karatas *et al.*, 2020)
ADA	CIC-IDS2017	0.999	0.999	0.999	This research
ADA	CSE-CIC-IDS2018	0.999	0.999	0.999	This research
ADA	LITNET-2020	0.997	0.996	0.997	This research
ID3	CIC-IDS-2017	0.98	0.98	0.98	(Sharafaldin *et al.*, 2018)
DT	CSE-CIC-IDS2018	0.997	0.997	0.997	(Karatas *et al.*, 2020)
DT	CSE-CIC-IDS2018	0.999	0.999	0.999	(Kilincer *et al.*, 2021)
CART	CIC-IDS2017	0.997	0.997	0.997	This research
CART	CSE-CIC-IDS2018	0.997	0.998	0.998	This research
CART	LITNET-2020	0.995	0.985	0.995	This research
GBC	CSE-CIC-IDS2018	0.995	0.991	0.993	(Karatas *et al.*, 2020)
GBC	CIC-IDS2017	0.997	0.997	0.997	This research
GBC	CSE-CIC-IDS2018	0.970	0.961	0.965	This research
GBC	LITNET-2020	0.987	0.756	0.987	This research
KNN	CIC-IDS-2017	0.96	0.96	0.96	(Sharafaldin *et al.*, 2018)
KNN	CSE-CIC-IDS2018	0.998	0.999	0.998	(Kanimozhi and Jacob, 2019a)
KNN	CSE-CIC-IDS2018	0.993	0.985	0.979	(Karatas *et al.*, 2020)
KNN	CSE-CIC-IDS2018	0.958	0.958	0.955	(Kilincer *et al.*, 2021)
KNN	CIC-IDS2017	0.994	0.994	0.994	This research
KNN	CSE-CIC-IDS2018	0.989	0.989	0.985	This research
KNN	LITNET-2020	0.957	0.864	0.955	This research
MLP	CIC-IDS-2017	0.77	0.83	0.76	(Sharafaldin *et al.*, 2018)
MLP	CSE-CIC-IDS2018	1.0	1.0	1.0	(Kanimozhi and Jacob, 2019a)
MLP	CIC-IDS2017	0.981	0.980	0.980	This research
MLP	CSE-CIC-IDS2018	0.960	0.959	0.958	This research
MLP	LITNET-2020	0.933	0.698	0.929	This research
LSTM	CSE-CIC-IDS2018	1.0	1.0	1.0	Dutta *et al.* (2020)
DNN	CSE-CIC-IDS2018	1.0	1.0	1.0	Dutta *et al.* (2020)
QDA	CIC-IDS-2017	0.97	0.88	0.92	(Sharafaldin *et al.*, 2018)
LDA	CSE-CIC-IDS2018	0.989	0.991	0.990	(Karatas *et al.*, 2020)
QDA	CIC-IDS2017	0.966	0.932	0.944	This research
QDA	CSE-CIC-IDS2018	0.712	0.648	0.597	This research
QDA	LITNET-2020	0.980	0.992	0.979	This research
RFC	CIC-IDS-2017	0.98	0.97	0.97	(Sharafaldin *et al.*, 2018)
RFC	CIC-IDS-2017	0.999	0.999	0.999	(Sharafaldin *et al.*, 2019)
RFC	CSE-CIC-IDS2018	0.999	0.999	0.999	(Kanimozhi and Jacob, 2019a)
RFC	CSE-CIC-IDS2018	0.993	0.992	0.993	(Karatas *et al.*, 2020)
RFC	CIC-IDS2017	0.998	0.998	0.998	This research
RFC	CSE-CIC-IDS2018	0.991	0.993	0.992	This research
RFC	LITNET-2020	0. 996	0.997	0.996	This research

[1] See explanatory notes related to cited work in Section 5.2.

classification problems, therefore, produces results that are incomparable to our research results. Features in their research are also selected differently, first utilizing Primary Components Analysis (PCA), then the Ensemble Feature Selection (EFS), using *EFS Package* in *R Studio* and ensemble methods *gbm*, *glm*, *lasso*, *ridge and treebag* from the *fscaret*

library. The AdaBoost classification with default weak decision tree classifiers was used during the training. Meanwhile, in our research a choice was made to strengthen the base classifier via pruning. The results of *Precision*, *Recall* and F_1 obtained are represented in Table 19.

Kanimozhi and Jacob (2019a, 2019b) classified the CSE-CIC-IDS2018 data set using ADA, RF, kNN, SVM, NB and ANN (Artificial neural network) machine learning methods. For an ANN authors used MLP with two layers, *lbfgs* solver, grid searched *alpha* parameter (for L2 regularization) and *Hidden layer* sizes. In their research, authors used 0–1 classification. Either "Benign" or "Malicious" labels were used for training, making the results directly incomparable with our multi-class approach. Results of the *accuracy*, *precision*, *recall*, F_1 and *AUC* were obtained. The results of *Precision*, *Recall* and F_1 are represented in Table 19.

In the study Karatas *et al.* (2020) classified the CSE-CIC-IDS2018 dataset using KNN, RFT, GBC, ADA, DT (Decision tree), and LDA (Linear discriminant analysis with singular value decomposition solver) algorithms. Parameters that were selected for all the implemented algorithms are described in Karatas *et al.* (2020) Table 8. Number of classes was determined to be six (one for non-attack type, and 5 for attack types), making the results directly incomparable with our multi-class approach. Cross-validation with 80%/20% split of training and test data was used. Results of the *accuracy, precision, recall* and F_1 were obtained. The results of *Precision*, *Recall* and F_1 are represented in Table 19.

In their study Kilincer *et al.* (2021) classified the CSE-CIC-IDS2018 dataset using KNN, DT, and SVM algorithms. Options of Matlab for KNN with KNN Fine algorithm, DT with Fine tree and SVM Quadratic algorithm gave the best results in this research. Results on a limited amount of records (up to 1584 records per class, see Kilincer *et al.* (2021) Table 3) were used in this research for CSE-CIC-IDS2018 dataset classes. Authors focus on UNSW-NB15 dataset with no discussion on pre-processing for CSE-CIC-IDS2018, parameter search or tree pruning or overfitting. Results of the *accuracy, precision, recall*, F_1 and *g-mean* were obtained. The results of *Precision*, *Recall* and F_1 are represented in Table 19.

In Dutta *et al.* (2020) authors used SMOTE and ENN to balance the LITNET-2020 dataset. Classes are reduced to two, *normal* and *malignant*, therefore, results are directly incomparable with ours. The approach also differs in that authors reduce dimensionality with Deep sparse autoencoder (Zhang *et al.*, 2018), selecting 15 features. Then authors stack LSTM with *adam* optimizer and DNN with four layers, back-propagation and stochastic gradient descent as the optimizer and early stopping on Keras with TF backend and Scikit-learn. 5-fold validation was used in that research. Results of the precision, recall, false positive rate, and MCC were obtained. The results of *Precision*, *Recall* and F_1 are represented in Table 19.

5.3. *Known Limitations*

Regarding the limitations of the approach taken in this research, it is important to note that new categories of malicious traffic in reality are introduced daily. Therefore, models tuned using this method will not detect zero day threats.

Table 20
MLP model results for Precision (Pr), and G-mean ($\bar{G}$) on LITNET-2020 dataset before and after SMOTE.

Class[1]	nPr	sPr	$n\bar{G}$	$s\bar{G}$
Reaper Worm	0	0.778	0	0.972
Spam Botnet	0.631	0.912	0.766	0.988
W32.Blaster	0	0.285	0	0.969

[1]Selected example rare classes.

Another know limitation is that in *absolute rarity* case, or when data has not been obtained and labelled sufficiently, models will predict with high *Error rate*. A possible known solution to this problem is an anomaly detection for the unseen data.

Moreover, datasets CIC-IDS2017 and IDS-2018 lack some categorical flag data, which is possible to obtain, like it has been demonstrated in LITNET-2020 case.

Even though LITNET-2020 lacks temporal features, introduced in CIC-IDS datasets, this, however, can be resolved by running the CICFlowMeter on the original PCAP files.

Temporal average approach of flags does not help some classes like *Infiltration*, however, flag features could be added to CIC-IDS datasets in the future.

While SMOTE was helpful for some rare classes, the method did not help much where sub-classes overlap due to lack of host data or feature latency.

Some features can be extracted and supplemented, which might be used in future research, however, extraction requires high degree of previous network traffic logging, whereas authors are aware that organizations lack resources to collect data on such a level of detail.

5.4. *Observations on Multi-Class Predictions*

Details of comparison of each class and dataset before and after SMOTE up-sampling is not represented here due to substantial amount of tables. However, it is important to note that some rare classes in these datasets learn very well even with a small numbers of records, which is confirmed by testing using dedicated unseen data. Some classes learn significantly better after adding synthetic data, which is further supported with tests on model performance and classification reports executed before (prefixed with n as nPr and $n\bar{G}$ for no-SMOTE) and after enriching data using SMOTE procedure in Table 20 prefixed with s as sPr and $s\bar{G}$.

As demonstrated in Table 20, random data under-sampling and SMOTE over-sampling techniques are supportive in ensuring that extremely under-represented classes (see Table 4) can learn with non-zero precision and $\bar{G}$, or provide better results.

6. Conclusions

In this paper, we have studied three highly imbalanced network intrusion datasets and proposed methodology steps (see Section 4), helping to achieve high classification results

of rare classes which were validated through model error decomposition and 50% data hold-out strategy. This methodology was checked using a novel, differently structured dataset LITNET-2020, and comparison of the results to those obtained on the established benchmark datasets CIC-IDS2017 and CSE-CIC-IDS2018.

A review of the LITNET-2020 dataset compliance to the criteria raised by Gharib *et al.* (2016) is first introduced in Section 2.2. A variant of random under-sampling (skewed ratio under-sampling, proposed by authors and discussed in Section 3.1), is used to reduce imbalance of classes in a nonlinear fashion, and SMOTE-NC up-sampling (see Section 3.2) is executed to increase representation of under-represented classes. Further on in this research, comparison of multi-class classification performance of the CIC-IDS2017 and CIC-IDS2018 datasets with the recent LITNET-2020 dataset is discussed in Section 5. As LITNET-2020 is constructed differently from the CIC-IDS datasets, a conclusion can be made that the proposed method is resistant to dataset change. Performance metrics – balanced accuracy (Formula (2)) and geometric mean of recall (Formula (4)), better suited for multi-class classification used for the LITNET-2020 dataset, is another introduced novelty (see results in Tables 16 and 17), not discussed by other authors using these datasets. Multi-criteria scoring is cross-validated with an approach of testing through data previously unseen for the models (see Section 4). Additional ML model, Gradient Boosting Classifier, utilizing ensemble of classification and regression trees, was introduced for benchmark in this research via the use of *XGBoost* library (Chen and Guestrin, 2016) incarnation with GPU support (see Section 3.5.6). In our methodology, cost sensitive model implementations have been used and have provided some better results (see Table 19) compared to other reviewed studies. Furthermore, selection of models with better generalization capabilities in this research has been achieved through decomposition of classification error into bias and variance (see results in Table 18). Instead of the weak *CART* base classifiers (see Section 3.8) parameters were GirdSearch'ed and parameters *Tree depth* and *alpha* were validated using the method of maximum cost path analysis (Breiman *et al.*, 1984). Other models were tuned using *Gridsearch* and *Balanced Accuracy Score* was scored as an optimization goal.

Machine learning algorithm rankings based on *Precision*, *Balanced Accuracy Score*, $\bar{G}$, and *Bias – Variance decomposition of Error*, show that tree ensembles (*Adaboost*, *Random Forest Trees and Gradient Boosting Classifier*) perform best on the compared here network intrusion datasets, including the recent LITNET-2020.

References

Adomavicius, G., Kwon, Y. (2011). Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5), 896–911.

Batista, G.E.A.P.A., Prati, R.C., Monard, M.C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*. https://doi.org/10.1145/1007730.1007735.

Breiman, L. (2001). Random forests. *Machine Learning*, 45, 58–32 https://doi.org/10.1023/A:1010933404324.

Breiman, L., Friedman, J., Stone, C., Olshen, R. (1984). *Classification and Regression Trees (Wadsworth Statistics/Probability)*, 0412048418. CRC Press, New York,

Brownlee, J. (2020). *Imbalanced Classification with Python – Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning*. Machine Learning Mastery, San Juan, pp. 463.

Buczak, A., Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys I& Tutorials*, 18, 1153–1176. https://doi.org/10.1109/COMST.2015.2494502.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. https://doi.org/10.1613/jair.953.

Chen, T., Guestrin, C. (2016). XGBoost: a scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*. ACM, New York, NY, USA, pp. 785–794. 978-1-4503-4232-2. https://doi.org/10.1145/2939672.2939785.

Chicco, D., Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1). https://doi.org/10.1186/s12864-019-6413-7.

Claise, B. (2004). *RFC 3954, Cisco Systems NetFlow Services Export Version 9*. Technical report, IETF. https://doi.org/10.17487/rfc3954.

Damasevicius, R., Venckauskas, A., Grigaliunas, S., Toldinas, J., Morkevicius, N., Aleliunas, T., Smuikys, P. (2020). Litnet-2020: An annotated real-world network flow dataset for network intrusion detection. *Electronics (Switzerland)*, 9(5). https://doi.org/10.3390/electronics9050800.

Domingos, P. (2000). A unified bias-variance decomposition and its applications. In: *Icml*, pp. 231–238. 2065432969.

Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., Ghorbani, A.A. (2016). Characterization of encrypted and VPN traffic using time-related features. In: *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*, PP. 407–414. https://doi.org/10.5220/0005740704070414.

Dudani, S.A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, pp. 325–327. https://doi.org/10.1109/TSMC.1976.5408784.

Dutta, V., Choraś, M., Pawlicki, M., Kozik, R. (2020). A deep learning ensemble for network anomaly and cyber-attack detection. *Sensors (Switzerland)*, 20(16), 1–20. https://doi.org/10.3390/s20164583.

Ferri, C., Hernández-Orallo, J., Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1), 27–38. https://doi.org/10.1016/j.patrec.2008.08.010.

Fisher, R. (1954). The analysis of variance with various binomial transformations. *Biometrics*, 10(1), 130–139.

Freund, Y., Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 119–139. https://doi.org/10.1006/jcss.1997.1504.

Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451.

Friedman, J.H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4), 367–378. https://doi.org/10.1016/S0167-9473(01)00065-2.

Garcia, V., Mollineda, R.A., Sanchez, J.S. (2010). Theoretical analysis of a performance measure for imbalanced data. In: *2010 20th International Conference on Pattern Recognition*. IEEE, Istanbul, pp. 617–620. 978-1-4244-7542-1. https://doi.org/10.1109/ICPR.2010.156.

Geisser, S. (1964). Posterior odds for multivariate normal classifications. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(1), 69–76.

Gharib, A., Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A. (2016). An evaluation framework for intrusion detection dataset. In: *2016 International Conference on Information Science and Security (ICISS)*. IEEE, Pattaya, Thailand, pp. 1–6. 978-1-5090-5493-0. https://doi.org/10.1109/ICISSEC.2016.7885840.

Hart, P.E. (1968). The condensed nearest neighbor rule (Corresp.). *IEEE Transactions on Information Theory*, 14(3), 515–516. https://doi.org/10.1109/TIT.1968.1054155.

He, H., Ma, Y. (2013). *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley, Piscataway, NJ, pp. 216. 9781118074626. https://doi.org/10.1002/9781118646106.

Hettich, S., Bay, S.D. (1999). The UCI KDD Archive http://kdd.ics.uci.edu. University of California, Department of Information and Computer Science.

Jurman, G., Riccadonna, S., Furlanello, C. (2012). A comparison of MCC and CEN error measures in multi-class prediction. *PLoS ONE*, 7(8), 41882. https://doi.org/10.1371/journal.pone.0041882.

Kanimozhi, V., Jacob, D.T.P. (2019a). Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. *International Journal of Engineering Applied Sciences and Technology*, 04(06), 209–213. https://doi.org/10.33564/IJEAST.2019.v04i06.036.

Kanimozhi, V., Jacob, T.P. (2019b). Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. *ICT Express*, 5(3), 211–214. 9781538675953. https://doi.org/10.1016/j.icte.2019.03.003.

Karatas, G., Demir, O., Sahingoz, O.K. (2020). Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. *IEEE Access*, 8, 32150–32162. https://doi.org/10.1109/ACCESS.2020.2973219.

Kilincer, I.F., Ertam, F., Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: datasets and comparative study. *Computer Networks*, 188(January), 107840. https://doi.org/10.1016/j.comnet.2021.107840.

Koch, R. (2011). Towards next-generation intrusion detection. In: *2011 3rd International Conference on Cyber Conflict*, pp. 151–168.

Kubat, M., Matwin, S. (1997). Addressing the curse of imbalanced data sets: one-sided sampling. In: *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186. https://doi.org/10.1007/3-540-62858-4_79.

Kurniabudi, Stiawan, D., Darmawijoyo, Bin Idris, M.Y.B., Bamhdi, A.M., Budiarto, R. (2020). CICIDS-2017 dataset feature analysis with information gain for anomaly detection. In: *IEEE Access*, pp. 132911–132921 https://doi.org/10.1109/ACCESS.2020.3009843.

Lashkari, A.H., Gil, G.D., Mamun, M.S.I., Ghorbani, A.A. (2017). Characterization of tor traffic using time based features. In: *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, pp. 253–262. 978-989-758-209-7. https://doi.org/10.5220/0006105602530262.

Laurikkala, J. (2001). *Improving Identification of Difficult Small Classes by Balancing Class Distribution*. Springer. 3540422943. https://doi.org/10.1007/3-540-48229-6_9.

LaValle, S.M., Branicky, M.S., Lindemann, S.R. (2004). On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7–8), 673–692.

Lawrence Berkeley National Laboratory (2010). *The Internet Traffic Archive*. http://ita.ee.lbl.gov/index.html.

Lemaitre, G., Nogueira, F., Aridas, C.K. (2016). Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18, 1–5.

Lemaître, G., Nogueira, F., Aridas, C.K. (2017). Imbalanced-learn: a python toolbox to tackle the urse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1–5.

Lin, D., Foster, D.P., Ungar, L.H. (2011). VIF regression: a fast regression algorithm for large data. *Journal of the American Statistical Association*, 106(493), 232–247. https://doi.org/10.1198/jasa.2011.tm10113.

Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A. (1999). Evaluating intrusion detection systems without attacking your friends: the 1998 DARPA intrusion detection evaluation. In: *Proceedings DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00*, PP. 12–26. https://doi.org/10.1109/DISCEX.2000.821506.

Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., Therón, R. (2018). UGR'16: a new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers and Security*, 73, 411–424. https://doi.org/10.1016/j.cose.2017.11.004.

Małowidzki, M., Berezinski, P., Mazur, M. (2015). Network intrusion detection: Half a kingdom for a good dataset. In: *Proceedings of NATO STO SAS-139 Workshop, Portugal*.

Matthews, B.W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) – Protein Structure*, 405(2), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9.

Mosley, L. (2013). *A balanced approach to the multi-class imbalance problem*. Iowa State University, Ames, Iowa. https://doi.org/10.31274/etd-180810-3375.

Ortigosa-Hernández, J., Inza, I., Lozano, J.A. (2017). Measuring the class-imbalance extent of multi-class problems. *Pattern Recognition Letters*, 98, 32–38. https://doi.org/10.1016/j.patrec.2017.08.002.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.

Raschka, S. (2018). *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*. http://arxiv.org/abs/1811.12808.

Ring, M., Wunderlich, S., Grudl, D. (2017). *Technical Report CIDDS-001 data set*, *001*, pp. 1–13.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security*, 86, 147–167. https://doi.org/10.1016/j.cose.2019.06.005.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton*. Cornell Aeronautical Laboratory.

Rosenblatt, F. (1962). *Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington.

Ross, B.C. (2014). Mutual information between discrete and continuous data sets. *PLoS ONE*, 9(2), 87357. https://doi.org/10.1371/journal.pone.0087357.

Seabold, S., Perktold, J. (2010). Statsmodels: econometric and statistical modeling with python. In: *9th Python in Science Conference*.

Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A. (2019). A detailed analysis of the CICIDS2017 data set. In: Mori, P., Furnell, S., Camp, O. (Eds.), *Information Systems Security and Privacy*. Springer International Publishing, Cham, pp. 172–188. 978-3-030-25109-3.

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Vol. 1. ICISSP, Funchal, Madeira, Portugal, pp. 108–116. 978-989-758-282-0. https://doi.org/10.5220/0006639801080116.

Shetye, A. (2019). *Feature Selection with Sklearn and Pandas*. https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b.

Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3), 357–374. https://doi.org/10.1016/J.COSE.2011.12.012.

Smith, M.R., Martinez, T., Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine Learning*, 95(2), 225–256. https://doi.org/10.1007/s10994-013-5422-z.

Sokolova, M., Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45, 427–437. https://doi.org/10.1016/j.ipm.2009.03.002.

Thakkar, A., Lohiya, R. (2020). A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167(2019), 636–645. https://doi.org/10.1016/j.procs.2020.03.330.

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*. https://doi.org/10.1016/j.aci.2018.08.003.

The Cooperative Association for Internet Data Analysis (2010). *CAIDA – The Cooperative Association for Internet Data Analysis*. http://www.caida.org/home/.

The Shmoo Group (2011). *Defcon*.

Tomek, I. (1976). Two modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics*. https://doi.org/10.1109/TSMC.1976.4309452.

Wei, J.M., Yuan, X.J., Hu, Q.H., Wang, S.Q. (2010). A novel measure for evaluating classifiers. *Expert Systems with Applications*, 37(5), 3799–3809. https://doi.org/10.1016/j.eswa.2009.11.040.

Wilson, D.L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3), 408–421. https://doi.org/10.1109/TSMC.1972.4309137.

Witten, I.H., Frank, E. (2002). Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1), 76–77. https://doi.org/10.1145/507338.507355.

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann Publishers, San Francisco, pp. 558. 0-12-088407-0.

Yulianto, A., Sukarno, P., Suwastika, N.A. (2019). Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset. *Journal of Physics: Conference Series*, 1192(1). https://doi.org/10.1088/1742-6596/1192/1/012018.

Zhang, C., Cheng, X., Liu, J., He, J., Liu, G. (2018). Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status. *Journal of Control Science and Engineering*, 1–9. https://doi.org/10.1155/2018/8676387.

V. Bulavas is a data privacy and information security officer at Vilnius University. His research interests include machine learning, information security and privacy. His academic background includes MSc in physics from Vilnius University and a MSc in public management from the Norwegian School of Management. He is certified with CISA, CGEIT, CRISC and CSM.

V. Marcinkevičius is a senior researcher, head of the Intelligent Technologies Research Group, and head of Artificial Intelligence Laboratory at Vilnius University, Institute of Data Science and Digital Technologies. His research interests include machine learning, information security and natural language processing. His academic background includes MSc in mathematics from Vilnius Educational University an PhD in informatics from Vytautas Magnus University.

J. Rumiński is a professor at Gdańsk University of Technology and a head of the Department of Biomedical Engineering, also a head of Gdańsk AI Bay club. His research interests include biomedical engineering and information security. His academic background includes MSc in medical devices, PhD in healthcare informatics and habilitation in biocybernetics and biomedical engineering from Gdańsk University of Technology.