Network Intrusion Detection Systems Design: A Machine Learning Approach

Manuel G. da Silva Neto, Danielo G. Gomes

[1]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Departamento de Engenharia de Teleinformática (DETI), Centro de Tecnologia, Campus do Pici
Universidade Federal do Ceará (UFC), Fortaleza - CE, 60455-970, Brazil

{manuelneto,dgomes}@great.ufc.br

Abstract. *With the increasing popularization of computer network-based technologies, security has become a daily concern, and intrusion detection systems (IDS) play an essential role in the supervision of computer networks. A current approach to detect network intrusions is the development of intrusion detection systems by employing machine learning techniques. Due to a variety of strategies used, there is a need for a systematic way that supports the decision making in a machine learning-based IDS project. In this paper, we present a systematic approach to decision-making support for algorithms selection on the IDS design. We used a very recent dataset and reduced their features from 78 to 51 through the mean decrease in impurity (MDI) feature selection technique. Afterward, we evaluated the network intrusion detection performance of eight machine learning algorithms on two dataset resampling techniques. Decision Trees, Random Forests and Multi-layer Perceptron on Stratified 10-Fold algorithms reached Precision, Recall, and F1-Scores metrics on about 98%-99% with low test times.*

1. Introduction

Intrusion Detection Systems (IDS) are systems built to monitor and analyze network traffic and hence detect anomalies and attack intrusions [Hindy et al. 2018]. From a machine learning perspective, the network intrusion detection can be considered a typical classification problem and the IDS project based on machine learning generally consists of three phases: (*i*) pre-processing, (*ii*) training, and (*iii*) detection [Li et al. 2019]. The project phases are composed of tasks that may require decision making by a strategy or technology. The selection of machine learning algorithms has its own set of activities and decision-making, as shown in figure 1.

Under a multidisciplinary view, as a classification problem, some additional challenges interfere with the design of an IDS. Traditional network intrusion detection systems have been developed using available knowledge bases (a.k.a. IDS datasets), and its performance depends heavily on the quality of the knowledge base [Li et al. 2019, Shiravi et al. 2012]. Therefore, the lack of an adequate public dataset severely impairs an IDS evaluation.

In this paper, we focus on algorithms selection decision-making tasks. We applied eight well-known machine learning algorithms to evaluate the accuracy of a general IDS based on the recent CICIDS2017 dataset [Sharafaldin et al. 2018]. Our contribution is threefold. First, we highlighted the decision-making viewpoint throughout the process

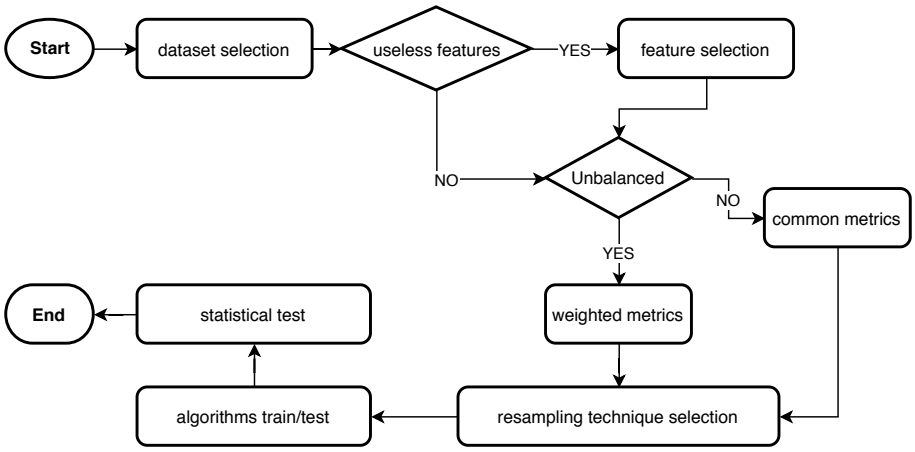

Figure 1. Algorithm selection decision-making activities.

and detailed all steps used for machine learning algorithms evaluation. Then, we evaluated the performance of the classifiers directly from the whole dataset with all attack labels, which is the most realistic way to deal with unseen data. Finally, we used two data resampling techniques to evaluate the intrusion detection performance of the classifiers on the selected feature set.

The remainder of the paper is organized as follows. Section 2 presents relevant related works. Section 3 shows some background for a better understanding of our proposal. Section 4 presents the proposed methodology, followed by the obtained results in Section 5 and discussion in Section 6. Finally, we present the conclusions in Section 7.

2. Related Works

Previous works have addressed, at different levels, the decision-making for algorithms selection on machine learning-based IDS design. This section presents some of these recent works by focusing on feature selection (FS) techniques, datasets, metrics, machine learning algorithms, and data resampling techniques used.

Varguese and Muniyal [Varghese and Muniyal 2017] studied the efficacy of seven different algorithms concerning two different feature selection strategies on the NSL-KDD dataset. They used Correlation-based Feature Selection (CFS) and Principal Component Analysis (PCA) for feature selection. They also evaluated the performance of j48, NBTree, Random Forest, LibSVM, Bagging with REPTree, PART, and Multilayer Perceptron(MLP) classifiers by using 10-Fold cross-validation. The authors evaluated the performance of the classifiers concerning feature extraction by calculating performance measures like accuracy, erroneous rate, recall, precision, F-measure, ROC and execution time.

Effendy *et al.* [Effendy et al. 2017] also used the NSL-KDD dataset and Information Gain Ratio (IGR) for feature selection. The authors evaluated the Naive-Bayes classifier with the accuracy as key performance indicator.

Biswas [Biswa 2018] studied the combination of feature selection techniques and classifiers to build accurate network intrusion detection. He applied four feature selection methods on the NSL-KDD dataset and evaluated the performance of five classifiers. The

author used 5-fold cross-validation and adopted the accuracy as the performance metric.

Park *et al.* [Park et al. 2018] evaluated the performance of detecting different types of attacks on Kyoto 2006+ dataset. They performed a selection of features manually, based on the recommendation of other works. The authors evaluated the Randon Forest classifier by using precision, recall, F1-Score, F2-Score, and Accuracy as the performance metrics. They manually separate the dataset from the training and test portions for each class label. They perform the training in the dataset and the metric estimation in the test subset only once.

Sharafaldin *et al.* [Sharafaldin et al. 2018] also extracted traffic features from the CICIDS2017 dataset and examined the performance and accuracy of the selected features with KNN, RF, ID3, Adaboost, MLP, Naive-Bayes, and QDA. They do not provide information about dataset splitting or resampling techniques usage on their classifiers performance evaluation.

Almseidin *et al.* [Almseidin et al. 2017] used an older dataset version, the KDD99. They evaluated the J48, Random Forest, Random Tree, Decision Table, MLP, Naive-Bayes, and Bayes Network classifiers by using precision, recall, ROC area, and Root Mean Squared Error as performance metrics. They do not provide information about dataset splitting and feature selection.

Ultimura and Costa [Utimura and Costa 2018] used 10% of the ISCXIDS2012 dataset to evaluate its algorithms. They manually separate the data in different training and test portions and perform the training and the metric estimation for each sub data proportion. The authors divided their work into two phases; the first one focused on the comparative analysis of the performance of the classifiers using the new ISCXIDS2012 dataset and the second on the validation of the SDI Snort ++ extension. The authors evaluate the Multilayer Perceptron (MLP) and the Optimum-Path Forest (OPF) classifiers by using the accuracy and execution time as performance metrics.

Table 1 summarizes the very recent works discussed where we highlight used dataset, feature selection, number of ML algorithms and split strategy. Here we used a new and still little-analyzed IDS dataset that contains the most current common attacks and evaluated the performance of network intrusion detection by adopting two data resampling techniques and eight classifiers on the entire dataset.

Table 1. Related works summary.

Reference	Dataset	FS	# algorithms	Split strategy
[Varghese and Muniyal 2017]	NSL-KDD	Yes	7	10-Fold
[Effendy et al. 2017]	NSL-KDD	Yes	1	not clear
[Almseidin et al. 2017]	KDD-99	No	7	not clear
[Biswa 2018]	NSL-KDD	Yes	5	5-Fold
[Park et al. 2018]	Kyoto 2006+	Yes	1	Train/test split
[Utimura and Costa 2018]	ISCXIDS2012	No	2	Train/test split
[Sharafaldin et al. 2018]	CICIDS2017	Yes	7	not clear
This paper	CICIDS2017	Yes	8	Strat. and 10-Fold

3. Background

3.1. CICIDS2017 Dataset

Sharafaldin *et al.* [Sharafaldin et al. 2018] analyzed the properties of eleven IDS datasets since 1998 and showed that most are out of date and unreliable. Some issues found are i) existing datasets suffer from the lack of traffic diversity and volumes, and ii) datasets do not cover the variety of known attacks.

CICIDS2017 dataset is for networking security and intrusion detection and is publicly available for research ends from Canadian Institute of Cyber-security [CIC 2018]. They captured data for a total of 5 days and extracted more than 80 network flow features from the generated network traffic (PCAP files). They also delivered the network flow dataset as CSV files which have 78 features and respective class labels. Implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. The CICIDS2017 dataset consists of labeled network flows, including full packet payloads in pcap format, the corresponding profiles and the labeled-flows and CSV files for the machine and deep learning purpose (please check Table 2).

Table 2. CICDS2017 dataset CSV.

File name	Classes
Monday-WorkingHours.pcap_ISCX.csv	Benign
Tuesday-WorkingHours.pcap_ISCX.csv	BruteForce, FTP-Patator, SSH-Patator, and Benign
Wednesday-workingHours.pcap_ISCX.csv	DoS/DDoS, DoSslowloris, DoSSlowhttptest, DoSHulk, DoSGoldenEye, Heartbleed, and Benign
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	WebAttack-BruteForce, WebAttack-XSS, WebAttack-Sql, and Benign
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Infiltration, MetaExploit, and Benign
Friday-WorkingHours-Morning.pcap_ISCX.csv	Botnet and Benign
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	DDoS and Benign
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	PortScan and Benign

3.2. Machine Learning on Network Intrusion Detection

Supervised learning approaches discover the patterns to map an input to an output based on the labeled input-output pairs of data samples. The classification problem is a typical supervised learning problem, which has been commonly used for network intrusion detection and many machine learning approaches have been used to solve network intrusion detection issues, and all of them consist of three general phases [Li et al. 2019]:

- **Preprocessing**: The data instances that are collected from the network environment are structured, which can then be directly fed into the machine learning algorithm. The processes of feature extraction and feature selection are also applied in this phase;
- **Training**: A machine learning algorithm is used to characterize the patterns of various types of data, and build a corresponding system model.

- **Detection**: Once the system model is built, the monitored traffic data will be used as system input to be compared to the system model.

Pedregosa *et al.* [Pedregosa et al. 2011] provides a brief explanation of evaluated algorithms as follows.

- **Nearest Centroid**: is a simple algorithm that represents each class by its member centroid. It has no parameters to choose, and for its simplicity, we choose it as our baseline algorithm;
- **Naive-Bayes**: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the *naive* assumption of conditional independence between every pair of features given the value of the class variable;
- **AdaBoost**: Its an ensemble boosting-based method. The core principle of AdaBoost is to fit a sequence of weak learners such as small decision trees, on repeatedly modified versions of the data and then combines the predictions through a weighted majority vote (or sum).
- **Multi-layer Perceptron**: MLP learns a mapping from input features to outputs and consists of layers of non-linear elements which form complex hypotheses.
- **Decision Trees**: are a non-parametric supervised learning method. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- **K-Nearest Neighbors**: is instance-based learning in which the algorithm does not attempt to construct a general internal model, but stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point.
- **Random Forests**: Use randomized decision trees as base learners. It is an ensemble bagging-based method in which the prediction is given as the averaged prediction of the individual base learners.
- **Quadratic Discriminant Analysis**: have closed-form solutions and no hyperparameters to tune. It has a quadratic decision surface and can be derived from simple probabilistic models.

Machine learning approaches have been susscessfully employed in the design of network intrusion detection systems as presented in the works of Park *et al.* [Park et al. 2018], Hodo *et al.* [Hodo et al. 2018], Shafaraldin *et al.* [Sharafaldin et al. 2018], Ultimura and Costa [Utimura and Costa 2018], and Biswa [Biswa 2018].

3.3. Resampling techniques

Regarding resampling techniques, a subset of samples are used to fit a model, and the remaining samples are used to estimate the efficacy of the model. This process is repeated many times, and the results are aggregated and summarized. These techniques for estimating model performance operate similarly, and their variations usually center around the method in which subsamples are chosen [Kuhn and Johnson 2013].

- **K-Fold Cross-validation**: The dataset samples are randomly partitioned into k sets of roughly equal size. Each partition, in turn, is used for testing and the remainder is used for training and repeat the procedure K times so that in the end, every partition has been used exactly once for testing. Finally, the K error estimates are averaged to yield an overall error estimate [Witten et al. 2017].

- **Stratified K-Fold Cross-validation**: A variant of K-Fold that select the k partitions in a way that makes the folds balanced concerning the outcome. Stratified K-Fold ensures that the sampling is done in a way that guarantees that each class is properly represented in both training and test sets [Kuhn and Johnson 2013].

Our work compares the network intrusion detection performance of the classifiers on CICIDS2017 by using these two resampling techniques.

4. Material and Methods

This section presents the adopted methodology and experiment details as shown in Figure 2. We divide our experiments into two scenarios, A and B. In experiment A, the focus was to evaluate the network intrusion detection performance of the classifiers in the CICIDS2017 dataset in a replicable and comprehensive manner. In experiment B the focus was to analyze whether a change in the resampling technique significantly alters the results of experiment A. We present the details of each experiment as follows.

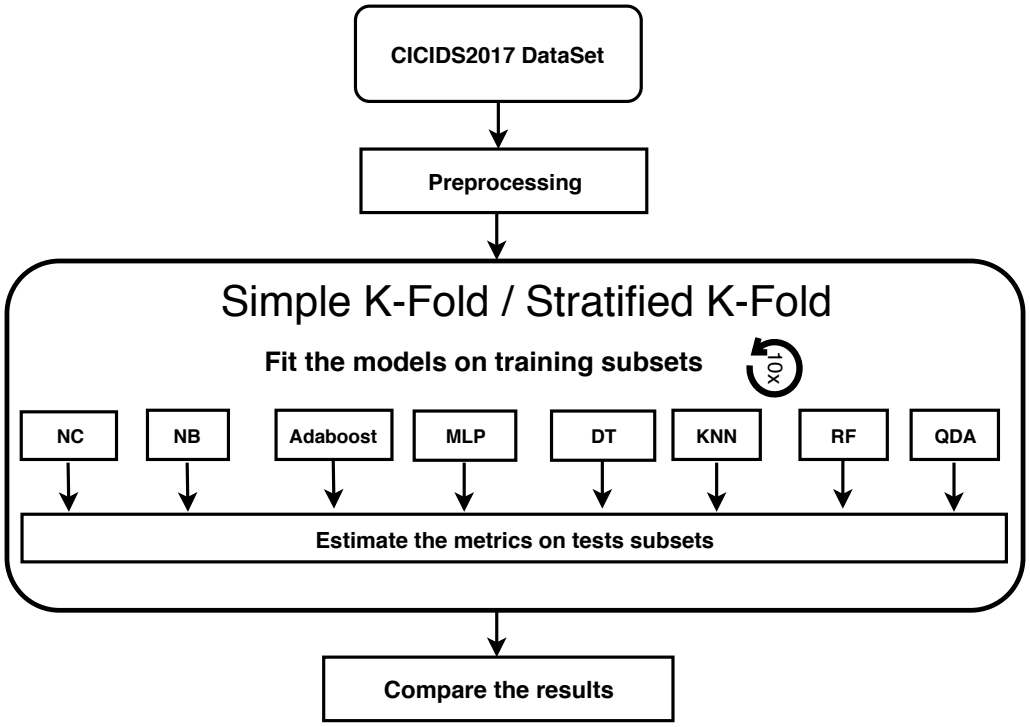

Figure 2. Experiments workflow.

Our experiments consisting into data preprocessing, apply the resampling technique for splitting the dataset into training and testing subsets. Within resampling, use the training subsets to fit each classifier and used the trained classifiers for predicting the labels of the test data subset. We compute the performance metrics of each trained classifier on the test subsets by comparing the actual labels and predicted labels. Finally, we summarize and discuss the results. The main difference between experiment A and B was the resampling technique adopted for dividing the dataset and evaluate the models. In experiment A we adopted 10-Fold cross-validation, and in experiment B we adopted Stratified 10-Fold cross-validation.

In this paper, the experiments where performed on Ubuntu Linux 18.04 LTS platform, Intel R, Core(TM) i7-7500U CPU 2.70GHz (4CPUs), 16 GB RAM. We employed Scikit-learn 0.20, Keras 2.2.0, and Python 3.6.5 on the classifiers implementation and evaluation. We also used the Rpy2 2.9.5 to perform statistics. Scikit-learn [Pedregosa et al. 2011] is a open source machine learning library and toolbox written in Python[1]. Keras[2] is a high-level neural networks API, also written in Python. Rpy2[3] is a Python interface to the R language.

The CICIDS2017 dataset is available as multiple CSV files, our preprocessing started by merging the files into a single CSV, removing strange characters from the column names and convert all non-numerical data into numerical representations. We decided to exclude as noise, records in which the word *Infinity* was amid the numerical values. For feature selection, we adopt Random Forests mean decrease in impurity (MDI) method directly on the whole dataset. By averaging the estimates of predictive ability over several randomized trees one can reduce the variance of such an estimate and use it for feature selection. This is known as the mean decrease in impurity [Pedregosa et al. 2011]. We end the preprocessing by applying the min-max data transformation that scales and transforms each feature individually such that it is in a small specified range as presented by Kuhn and Johnson [Kuhn and Johnson 2013].

Our evaluated classifiers are Randon Forests (RF), Naive-Bayes (NB), Adaboost, Multilayer Perceptron (MLP), Decision Trees (CART), K-Nearest Neighbors (KNN), Quadratic Discriminant Analysis (QDA), and our baseline for performance evaluation, the Nearest Centroid (NC) classifier. We configure our MLP as 51/102/51/15. This setup has 51 neurons in input layer that represent each feature, 102 neurons on the first hidden layer, 51 neurons on the second hidden layer, and finally, 15 neurons in the output layer that represents each class label. We used the *relu* activation function on hidden layers and *softmax* activation function in the output layer as Keras documentation recommendations. We adopted Scikit-learn default hyperparameters on the remaining classifiers.

The CICIDS2017 dataset has multiple attack labels, and we reframed it as a multiclass classification problem. We used the weighted averaged of the **Pr**, **Rc** and **F1** metrics as presented by Shafaraldin *et al.* [Sharafaldin et al. 2018].

Varguese and Muniyal [Varghese and Muniyal 2017] explain these metrics in a network intrusion detection way, as follow.

- **Precision (Pr)**: Precision or Predictive value positive is the proportion of positives (alerts) that corresponds to the presence of the attack condition.
- **Recall (Rc)**: Recall or Sensitivity is a true positive rate which measures the ability of a test to detect the attack condition when the attack condition is present.
- **F1-Score (F1)**: It is a harmonic combination of the Pr and Rc into a single measure. It adopts a weighted version of these metrics for multiclass classification problems.
- **Time to test**: The recognition of unseen data is essential on IDS design. Thus, we compute the time needed to run the already trained classifiers on the test set.

[1] `https://www.python.org/`
[2] `https://keras.io`
[3] `https://pypi.org/project/rpy2/`

Pedregosa *et al.* [Pedregosa et al. 2011] formally present Pr, Rc, and F1 as follows. If y is the set of predicted *(sample, label)* pairs. $\hat{y}$ is the set of true *(sample, label)* pairs. L is the set of labels. y_l the subset of y with label l and $\hat{y}_l$ is the subset of $\hat{y}$.

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN} \tag{1}$$

$$Pr_{weighted}(y, \hat{y}) = \frac{1}{\sum_{l \in L} |\hat{y}_i|} \sum_{l \in L} |\hat{y}_l| P(y_l, \hat{y}_l) \tag{2}$$

$$R_{weighted}(y, \hat{y}) = \frac{1}{\sum_{l \in L} |\hat{y}_i|} \sum_{l \in L} |\hat{y}_l| R(y_l, \hat{y}_l) \tag{3}$$

$$F1(y, \hat{y}) = \frac{2 * (precision * recall)}{precision + recall} \tag{4}$$

Within each resample type, we trained and tested the classifiers and computed the mean and standard deviation of the metrics for the set of ten rounds in experiments A and B. We used 10-Fold cross-validation in experiment A and Stratified 10-fold cross-validation in experiment B.

5. Results

5.1. Preprocess

Table 3 shows the final quantitative for each class label after noise clean up. We observe that the dataset is highly unbalanced.

Table 3. Dataset clean up

Class Name	Num. Repr.	Original Samples	Noise Samples	Final Samples
Benign	0	2273097	1777	2271320
Dos Hulk	4	231073	949	230124
PortScan	10	158930	126	158804
DDoS	2	128027	2	128025
Dos GoldenEye	3	10293	0	10293
FTP-Patator	7	7938	3	7935
SSH-Patator	11	5897	0	5897
DoS Slowloris	6	5796	0	5796
DoS Slowhttptest	5	5499	0	5499
Bot	1	1966	10	1956
Web Attack Brute Force	12	1507	0	1507
Web Attack XSS	14	652	0	652
Infiltration	9	36	0	36
Web Attack SQL Injection	13	21	0	21
Heartbleed	8	11	0	11
Total	-	2830743	2867	2827876

On feature selection, we compute the feature importance on the whole dataset (78 features and one class label column) by using MDI technique within the RandonForestRegressor class of scikit-learn [Pedregosa et al. 2011] as presented by [Sharafaldin et al.

2018]. However, due to a large number of label classes (15) with different characteristics to be identified, we decided to keep all features with non-zero importance, thus excluding only the features in which MDI indicated zero importance. Table 4 lists our 51 features selected. The final evaluated dataset has 51 feature columns and one column with class labels.

Table 4. Features selected

Features selected regarding the whole dataset - nonzero importance
Flow_Bytes/s, Total_Length_of_Fwd_Packets, Bwd_Packet_Length_Std, Subflow_Fwd_Bytes, Destination_Port, PSH_Flag_Count, Bwd_Packets/s, Flow_IAT_Mean, Fwd_Packet_Length_Std, Init_Win_bytes_forward, Fwd_IAT_Min, Init_Win_bytes_backward, min_seg_size_forward, Average_Packet_Size, Fwd_Header_Length.1, Fwd_Header_Length, Bwd_Packet_Length_Mean, Idle_Min, Fwd_Packet_Length_Max, Total_Length_of_Bwd_Packets, Subflow_Bwd_Bytes, Active_Std, Bwd_IAT_Mean, Bwd_IAT_Std, Packet_Length_Mean, Bwd_Packet_Length_Max, Bwd_IAT_Min, Avg_Bwd_Segment_Size, Flow_IAT_Std, Flow_IAT_Min, URG_Flag_Count, Min_Packet_Length, Max_Packet_Length, Fwd_Packet_Length_Min, Flow_Packets/s, Flow_IAT_Max, Flow_Duration, Bwd_Header_Length, Fwd_Packets/s, Fwd_Packet_Length_Mean, Fwd_IAT_Total, Fwd_IAT_Std, Fwd_IAT_Mean, Fwd_IAT_Max, act_data_pkt_fwd, Total_Fwd_Packets, Fwd_PSH_Flags, Down/Up_Ratio, Bwd_IAT_Max, Avg_Fwd_Segment_Size, ACK_Flag_Count

5.2. Machine Learning Classifiers Evaluation

This section presents experiment result details and comparison. Table 5 and table 6 shows the mean and standard deviation of our evaluation metrics for each classifier within a cross-validation (CV) evaluation.

5.2.1. Experiment A

An individual analysis of experiment A concerning each metric shows that the use of simple 10-Fold CV generated high standard deviation rates with inconstant results in each training round and tests as presented in table 5. We use the results of the precision, recall, and f1-score metrics of experiment A for comparison purposes with those of experiment B.

Table 5. Experiment A - Performance Evaluation Results

Experiment A - Simple K-fold								
Algorithm	Precision		Recall		F1-Score		Test time (sec.)	
	mean	std	mean	std	mean	std	mean	std
RF	0.84	0.27	0.88	0.20	0.86	0.25	1.32	0.08
Naive-Bayes	0.90	0.16	0.58	0.14	0.70	0.16	3.23	0.17
AdaBoost	0.76	0.30	0.82	0.22	0.78	0.27	21.26	1.10
MLP	0.83	0.29	0.88	0.19	0.85	0.26	2.05	0.22
DT	0.85	0.27	0.89	0.20	0.86	0.25	0.23	0.01
KNN	0.84	0.29	0.88	0.19	0.85	0.26	2411.17	381.32
NC	0.83	0.23	0.49	0.13	0.61	0.16	0.42	0.01
QDA	0.81	0.29	0.80	0.21	0.77	0.28	7.47	0.71

5.2.2. Experiment B

Table 6 depicts experiment B in which we used the Stratified 10-Fold CV. We observe low standard deviations on evaluation metrics that indicates low fluctuations around the mean in each round of training and test. Concerning precision, RF, MLP, and DT achieved 99%. KNN and QDA reached 98%. NB produced 97%, NC reached 84%, and the worst result was Adaboost with 75% of precision that was overcome even by the baseline classifier. Regarding recall, the best results were DT with 99% and RF with % 98. KNN and MLP achieved 97%. QDA reached 95% and Adaboost 85%. The worst result was of NB with 67% and the baseline NC with 55% recall. F1-score metric presents DT as the best result with 99 %. MLP and RF achieved 98 %. KNN reached 97% and QDA 96%. The worst results were NB with 78% and our baseline, NC with 63%. Since the F1-Score is the combination of recall and precision, most of those classifiers achieved over 80% that is good values.

On the testing times evaluation, the worst result was of KNN since this is an instance-based algorithm in which the amount of data harmed its average time of tests. DT and NC were the fastest. The remaining algorithms performed the tests on average in few seconds.

Table 6. Experiment B - Performance Evaluation Results

Experiment B - Stratified K-fold								
Algorithm	Precision		Recall		F1-Score		Test time (sec.)	
	mean	std	mean	std	mean	std	mean	std
RF	0.99	0.01	0.98	0.03	0.98	0.02	1.40	0.04
Naive-Bayes	0.97	0.01	0.67	0.04	0.78	0.04	5.83	0.74
AdaBoost	0.75	0.01	0.85	0.01	0.80	0.01	23.62	0.20
MLP	0.99	0.13	0.97	0.05	0.98	0.03	1.60	0.13
DT	0.99	0.008	0.99	0.01	0.99	0.01	0.23	0.006
KNN	0.98	0.01	0.97	0.06	0.97	0.04	2435.44	270.74
NC	0.84	0.04	0.55	0.07	0.63	0.06	0.42	0.006
QDA	0.98	0.007	0.95	0.01	0.96	0.01	12.25	0.10

5.2.3. Experiment A and B Comparison

The decrease in the standard deviations of experiment B regarding experiments A is perceptible. However, since the mean is sensitive to variations, it is relevant to test whether the set of results of a classifier in experiment B overcome the set of results of the same classifier in experiment A.

We applied the Wilcoxon matched-pairs signed-ranks test to compare the set of 10 CV results between experiment A and B of each classifier. The default assumption for the test is that the two samples have the same distribution. The Wilcoxon matched-pairs signed-ranks test is a nonparametric procedure that can be employed to evaluate a before-after design whenever one or more of the assumptions of t-test for two dependent samples are violated [Sheskin 2007]. Hypotheses being tested is:

- **Null hypothesis** ($H_0 = 0$): There is no significant difference between a classifier result and their pair in experiment A and B.
- **Alternative hypothesis** ($H_A < 0$): The classifier scores in Experiment B are higher than their pair scores in Experiment A. This is a directional alternative hypothesis and it is evaluated with a one-tailed test [Sheskin 2007].

The null hypothesis can only be rejected if the computed value T_W is equal to or less than the Wilcoxon tabled critical value at the prespecified level of significance [Sheskin 2007]. With ten repetitions (n=10), the table of critical T values for Wilcoxon's tests with one-tailed .10 level of significance is exactly $T_{.10} = 14$ [Ott and Longnecker 2015]. We tested the hypothesis for precision, recall, and f1-score of classifiers concerning experiment A and B which is shown in table 7.

Table 7. Wilcoxon Matched-Pairs Signed-Rank Tests - A and B comparison

T_W statistics for $n = 10$ and Critical $T_{.10} = 14$						
Algorithm	Precision		Recall		F1-Score	
	T_W	p-value	T_W	p-value	T_W	p-value
RF	14	0.09	14	0.09	14	0.09
Naive-Bayes	28	0.53	10	0.04	13	0.08
AdaBoost	28	0.53	28	0.53	28	0.53
MLP	16	0.13	13	0.08	14	0.09
DT	11	0.05	10	0.04	10	0.04
KNN	17	0.16	14	0.09	15	0.11
NC	33	0.72	18	0.18	35	0.78
QDA	20	0.24	8	0.02	15	0.11

The results showed that by using Stratified 10-Fold in experiment B, the Random Forest and Decision Tree overcome their pairs on Pr, Rc, and F1-score. Naive-Bayes and MLP overcome their pairs in Rc and F1-score. KNN and QDA overcome just in the recall. For $T_W > T_{.10}$ we cannot reject the null hypothesis and these results do not differ statistically between experiments A and B.

6. Discussion

In this section, we discuss the results and present the main threats to the validity of this work. As presented in Section 2, Shafaraldin *et al.* [Sharafaldin et al. 2018] also evaluated the performance of classifiers in their dataset as shown in table 8. They did not provide information on the percentage of the dataset used for training and testing or the number of test rounds. Comparing the results of Pr, Rc, and F1, our experiment B overcome their results in all metrics except for Adaboost classifier Pr which was our worst result. The most significant gains were for the Rc and F1 metrics.

Table 8. Shafaraldin *et al.* [Sharafaldin et al. 2018] Results

Algorithm	Pr	Rc	F1	Execution(sec.)
KNN	0.96	0.96	0.96	1908.23
RF	0.98	0.97	0.97	74.39
ID3	0.98	0.98	0.98	235.02
AdaBoost	0.77	0.84	0.77	1126.24
MLP	0.77	0.83	0.76	575.73
Naive-Bayes	0.88	0.04	0.04	14.77
QDA	0.97	0.88	0.92	18.79

By considering our experiment B performance metrics, the decision making for a classifier depends on IDS design needs. Thus, when the cost of false positives is hight, Pr is the best metric. When the cost of false negative is hight, Rc is the best. F1 combines these two metrics and is the best if there is a need for a balance between Pr and Rc. The testing time is also critical for IDS systems design so it should be taken into account.

We used F1 as the primary metric and Pr and Rc values as secondary for decision making. We also used the time metric when their value turns the classifier adoption impractical. In these criteria, the best performing classifiers were Decision Trees with 99%, Random Forest and Multi-layer Perceptron with 98%, and Quadratic Discriminant Analysis with 96%. The KNN classifier achieved hight F1, Pr, and Rc values but the testing time invalidates their results. The AdaBoost and Naive-Bayes present hight variations between Pr and Rc that decrease their F1 scores. Finally, the Nearest Centroid was our baseline and its hight dependant on data geometry. Despite the hight Pr value, the Rc and F1 for Nearest Centroid were low as expected.

6.1. Threats to validity

Since the CICIDS2017 is a new and few explored dataset, we made the experiment design decision of evaluating eight algorithms in the dataset in its totality, and this has some implications on algorithms comparison results and statistical test adopted. The comparative evaluation of algorithms in the detection of network intrusion is essential for the design of machine learning-based IDS. However, the literature presents different arguments on how to perform this comparison.

Dietterich [Dietterich 1998] recommends a paired-T tests classifiers comparison within a five times 2-fold cross-validation. Demšar [Demšar 2006] recommends using the Wilcoxon test since that non-parametric tests are suitable for classifiers comparison and safer than parametric tests that assume normal distributions or homogeneity of variance.

Witten *et al.* [Witten et al. 2017] claim that single 10-fold cross-validation might not be enough to get a reliable error estimate and when seeking an accurate error estimate with limited data, an alternative is to repeat the cross-validation process n times on dataset obtaining statistically reliable results.

Wilcoxon tests have exact values for n = 10 without normal approximations. Although our research designs computationally limited our experiments to one 10-fold execution, we believe that our experiment A and B comparison has statistical representativity.

7. Conclusions

We presented an approach for a machine learning-based IDS design to support the decision making for the classifier selection and the dataset resampling technique. Here we performed an experimental study of two classification scenarios to evaluate the network intrusion detection performance of eight machine learning algorithms on the CICIDS2017 dataset. We applied the Random Forests MDI method in selecting relevant features for attack classification by reducing the dataset from 78 to 51 columns. We also showed that the dataset is hight unbalanced. Afterward, we estimated the performance of RF, NB, AdaBoost, MLP, DT, KNN, NC, and QDA within 10-Fold and Stratified 10-Fold CV on CICIDS2017 dataset.

From our results about machine learning-based IDS design, we should take into account not only the dataset quality but also the resampling technique used for classifiers evaluation. In this paper, Stratified K-Fold achieved high scores and low variations than K-Fold. We used F1 as the key metric and Rc, Pr, and Time as secondary metrics concerning IDS design classifiers decision making. With this criteria, the best performing classifiers were Decision Trees, Random Forests and Multi-layer Perceptron. Finally, we showed that KNN classifier achieved competitive results but its testing time is prohibitive.

Our findings must be useful for network intrusion detection researchers and practitioners by supporting them on the IDS datasets and classifiers selection decision-making. Investigations will be performed in the future to explore the applicability of our selected classifiers in real-time intrusions detection.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES) - Finance Code 001. Danielo thanks the financial support of the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico-Brasil, processes #432585/2016-8, #311878/2016-4).

References

Almseidin, M., Alzubi, M., Kovacs, S., and Alkasassbeh, M. (2017). Evaluation of machine learning algorithms for intrusion detection system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 277–282.

Biswa, S. K. (2018). Intrusion detection using machine learning: A comparison study. *International Journal of Pure and Applied Mathematics (IJPAM)*, 118(19):101–114.

CIC (2018). Intrusion detection evaluation dataset (cicids2017). `https://www.unb.ca/cic/datasets/ids-2017.html`.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Effendy, D. A., Kusrini, K., and Sudarmawan, S. (2017). Classification of intrusion detection system (ids) based on computer network. In *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pages 90–94.

Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., and Bellekens, X. (2018). A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. *arXiv preprint arXiv:1806.03517*.

Hodo, E., Bellekens, X., Iorkyase, E., Hamilton, A., Tachtatzis, C., and Atkinson, R. (2018). Machine learning approach for detection of nontor traffic. *Journal of Cyber Security*, 6(2):171–194.

Kuhn, M. and Johnson, K. (2013). *Applied predictive modeling*, volume 26. Springer-Verlag New York.

Li, J., Qu, Y., Chao, F., Shum, H. P. H., Ho, E. S. L., and Yang, L. (2019). *Machine Learning Algorithms for Network Intrusion Detection*, pages 151–179. Springer International Publishing, Cham.

Ott, R. L. and Longnecker, M. T. (2015). *An Introduction to Statistical Methods and Data Analysis Sixth Edition*. Nelson Education.

Park, K., Song, Y., and Cheong, Y. (2018). Classification of attack types for intrusion detection systems using a machine learning algorithm. In *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 282–286.

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., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sharafaldin, I., Lashkari, A. H., and 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 (ICISSP)*, pages 108–116.

Sheskin, D. J. (2007). *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 4 edition.

Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357 – 374.

Utimura, L. N. and Costa, K. A. (2018). Aplicação e análise comparativa do desempenho de classificadores de padrões para o sistema de detecção de intrusão snort. In *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, volume 36.

Varghese, J. E. and Muniyal, B. (2017). An investigation of classification algorithms for intrusion detection system — a quantitative approach. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2045–2051.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2017). *Data Mining: Practical Machine Learning Tools and Techniques Fourth Edition*. Morgan Kaufmann.