Machine Learning Classifiers Selection in Network Intrusion Detection

Graciela Becci[1], Javier Díaz[2], Luis Marrone[2], Miguel Morandi[1]

[1] Universidad Nacional de San Juan, Av Libertador San Martín Oeste 1109, San Juan, J5400ARL Argentina
[2] Universidad Nacional de La Plata, Facultad de Informática, Calle 50 y Av 120, La Plata, Buenos Aires, B1900ASF Argentina
gbecci@unsj.edu.ar, jdiaz@unlp.edu.ar,
lmarrone@linti.unlp.edu.ar, morandi@unsj.edu.ar

Abstract. The objective of this work is to select machine learning classifiers for Network Intrusion Detection NIDS problems. The selection criterion is based upon the hyper-parameter variation, to evaluate and compare consistently the different models configuration. The models were trained and tested by cross-validation sharing the same dataset partitions. The hyper-parameter search was performed in two ways, exhaustive and randomized upon the structure of the classifier to get feasible results. The performance result was tested for significance according to the frequentist and Bayesian significance test. The Bayesian posterior distribution was further analyzed to extract information in support of the classifiers comparison. The selection of a machine learning classifier is not trivial and it heavily depends on the dataset and the problem of interest. In this experiment seven classes of machine learning classifiers were initially analyzed, from which only three classes were selected to perform cross-validation to get the final selection, Decision Tree, Random Forest, and Multilayer Perceptron Classifiers. This article explores a systematic and rigorous approach to assess and select NIDS classifiers further than selecting the performance scores.

Keywords: machine-learning classifiers, network intrusion detection, cross-validation.

1 Introduction

At the core of a Network Intrusion Detection Systems **NIDS** there is a two-fold problem, feature selection and flow classification. Machine Learning **ML** offers efficient solutions to both problems. This article is focused on the classifier selection and problems found during the ML classifiers comparison and performance evaluation.

Section 2 presents a revision of the main concepts required to understand the article development, Section 3 presents the methodology used in the classifiers comparison and selection, Section 4 present the results of the experiments, and Section 5 presents conclusions related to the experience of selecting classifiers in NIDS environment.

2 Antecedents

2.1 NIDS type according to threat knowledge

To understand the ways of detecting and classifying an attack it is important to know the types of common NIDS.

NIDS type according to threat knowledge. Network threats can be classified as unknown and well-known threats. The unknown threats are for example zero-day attacks, undefined attacks and in general attacks not related to well-identified vectors. The unknown attacks are mainly identified by the flow behaviour, while well-known attacks are detected by the analysis of their specification, statistical description, and protocols behaviour under attack. The knowledge in these attacks allow to include them in black lists to be distributed in signature-based NIDS such as Snort and Suricata [1], [2].

Machine Learning helps to create models representing a profile of normal activity to be compared with the network flow to detect deviations. Mishra et al. identify four IDS categories based on ML, derived from combining single classifier and multiple classifiers with all features and with a reduced set of features [3]. The multiple classifier architecture allows for hierarchical intrusion detection with the aim of improving the detection rate in a similar way as a Decision Tree classify the data following a hierarchy of successive rules.

NIDS in anomaly and network misuse detection. Unusual network behaviour is detected by abnormal network patterns described by outliers and out of range data. In ML NIDS approach, the learning process starts registering normal flow to determine normal flow profile to be compared to network flow and detect deviations from normal flow. This approach has good detection performance but high number of false positives [4]. ML approaches use Autoencoders, considered as deep Neural Networks due to the high number of fully connected nodes, using un-supervised learning for learning the main flow characteristics [5], [6].

Well-Known Threats. Attacks in signature-based IDS are registered in a database and distributed as black-lists containing the main characteristics of historic attacks [7]. This commonly used method has a good detection rate of the attack included in the list, but fails to detect new attacks, so is the importance of keeping the black lists updated to include new attacks.

Specification-based NIDS looks for protocol and services specification infringement, which might be an indication of network attack [8]. Examples of attack indicators are IP source and destination address and port, flags, tags, and payload description, which may indicate network intrusion [9].

Flow statistics analysis is used in IDS to detect variation in normal flow [4]. This method gives better results when applied in a particular point inside the perimeter such as a particular address or port, to avoid false positives.

IDS based in the stateful protocol analysis, this analysis keep track of the protocol different states within a determined session, and look for anomalies in the session [10]. This method is time consuming for the network administrator. However it gives good understanding of network behaviour helping to discriminate false positive. All these methods in the practice are combined for better performance of the network defense.

Homoliak et al. compared three classifiers Decision Tree, Support Vector Machine SVM and Naïve Bayes for detecting adversarial obfuscation attacks in NIDS, basically tunneling and non-payload obfuscation [11]. This work used 194 features, and concluded that decision tree classifier was the most robust, while SVM achieves the worst performance. Ahmad et al compared four ML techniques for NIDS, the machines were SVM-Linear, SVM-RBF (Radial Basis Function), Random Forest, and Extreme Learning Machine ELM [12]. The conclusion of the experiment is that ELM outperforms the other approaches. The evaluation finishes with the accuracy and recall percentages. Ahmad et al. evaluated four ML techniques within SDN security environment [13]. The classifiers compared are SVM, Naïve Bayes, Decision Trees, and Logistic Regression. The results were that SVM outperform the other methods in terms of accuracy and precision. No further details about the number of features are given. In these articles and in many cases no further support to the experiment is given in the form of statistical significance test.

A tutorial article Statistical comparison of models using grid search was provided with simulated 50 samples to illustrate the differences between the frequentist and Bayesian statistical analysis [14]. This article was an example to understand the difference between the two approaches and the importance of validating empirical research.

Revision of the main Machine Learning Classifiers used in this work

Decision Tree Classifier DTC. A Decision Tree is a structure of nodes, branches and leaves for making successive decisions until find a final result. Decision Trees are used in classification in case of a discrete output variable, or in regression making prediction of continuous output.

The tree structure is defined by supervised training using the information of the features in a sample [15]. This information is refined in successive branch partitioning, leading to a better adaptation of the tree to the training sample. The more refined the decisions the deeper the tree, and higher the risk of over-fitting, meaning the tree fails to generalize or to resolve different samples. The methods for avoiding over-fitting are for example controlling the maximum tree depth by pruning the tree or using modeling constraints in the parameters. Nodes with few samples are avoided with constrains in the minimum number of samples at leaf node. In the extreme case, a node representing only one sample implies a highly specialized tree that adds little to the interpretation of the whole dataset.

The algorithms in Decision Trees split the features considering the homogeneity or purity of the target class [16]. Gini impurity and Entropy are the measurements for splitting a node based on the homogeneity and information gain respectively [15].

Decision Trees are prone to bias in the output if the set of target class is unbalanced, that is the classes in the output are not equally represented. In this case the tree tends to better classify samples with target classes that are majority. To avoid this problem it is recommended to balance the dataset before constructing the tree model.

The structure of decision trees is highly sensitive to smaller changes in the input parameters, requiring sometimes building a completely new tree.

Random Forest Classifier. Random forest is an ensemble of decision trees to classify or predict the class to which the input sample belongs. The final decision is by a voting system averaging the results or votes of individual decision trees [17]. The structure of the trees is determined by supervised learning. Trees are trained with a set of features and samples to find the related target class.

The result of Random Forest tends to have high variance and slightly high bias [15]. More independent trees characteristic means lower variance in the results. Thus, the independence is achieved by introducing randomness in the process of modeling of trees in different ways. For instance, the structure of the decision trees is selected by training the trees with random sampling with replacement or bootstrap of the dataset [18]. Also randomness is introduced by selecting a random set of features per decision tree. The randomness leads to error cancellation in the result. This is achieved also by the voting system that allows for some erroneous results to be cancelled out. The algorithm for voting could be by direct vote for a target class or by voting the probability of occurrence of the target class, as it is the alternative offered by the Scikit-learn Machine Learning Libraries for Python [19].

Multilayer Perceptron. Multilayer perceptron is a class of neural network where the neurons or nodes are connected following a feed-forward strategy: the nodes are connected to nodes in the next neurons layer, unlike recursive networks where the nodes can be connected to nodes in the same layer [20]. The objective of the network is to optimize some objective or cost function, which evaluates the network error resulting from comparing the calculated output and the expected output. The result of the objective function is back-propagated to change the network weights and bias in order to diminish or reduce the output error. This process is the learning process and it is supervised because given the input the output calculated is compared to the desired or expected given output to obtain the network error. The learning function stops when the error shows no further reduction in successive training steps.

The activation function is the transforming function that transforms the input from the nodes to an output. The aggregated input from nodes weighted by the respective weight is transformed by the activation function to give the network output which is evaluated by the cost function or objective function which gives the network error that is back-propagated to the network nodes in order to change the nodes weights to reduce the output error and so on.

The activation function is one of the hyper-parameters of the network to set up before training; the selection criterion could be for example the convergence error-time. During the training process, the network converge to some value of the objective function and further training steps do not improve the objective or cost function it is time to stop the network training.

3 Methodology and Design of Experiment

Cross-validation for model selection

The cross-validation method looks for training and testing the model with separate datasets to improve model generalization, while avoiding over-fitting that happens when using the same training data for testing. Generalization means the model has good performance even with data not used for training. Over-fitting means that a model has a very good performance with data used for training but poor performance with data used for testing and validating.

The parameters of the model are selected during the training process. In the case of supervised learning it is possible to count with input-output pairs to verify the output of the trained model and derive the related error. This error is used as a feedback to adjust the parameters according to the selected training algorithm. In a second stage, the model performance is evaluated with the test set, and finally the validation of the model is evaluated by its capacity of making good predictions.

Large amount of good quality of data is fundamental for training good models. As this is an ideal condition in real experiments the design of experiment offer alternatives to draw train-test pairs from normal datasets. One of the simplest options is the random subsampling without replacement using the sample for training and the rest of data for testing, and repeating the process a number of times. Cross-validation is an alternative to get several train-test sets. The cross-validation method used in this experiment is the Repeated Stratified k-Fold, where the dataset is partitioned in k-folds consisting each of training and testing sets. The stratified term refers to its proportionality to the target class due to the unbalanced output, where the target classes are not equally represented.

Cross-validation Problem and Design of Experiment

Cross-validation may present overlapping of train-test sets. Overlapping means dependence between train-test sets and therefore co-variation between samples. From the frequentist approach overlapping or dependency between samples implies increment of type I and II error. Recalling that type I and II errors are indicators of the quality of a test. Type I error is the probability of false positive cases, which in the context of network intrusion detection, is the probability of mistakenly classifying a sample as an attack when in fact there is normal flow. Conversely, Type II error is the probability of false negative cases, which is the probability of mistakenly failing to classify an attack considering it as a normal flow.

Several experiments have been conducted with datasets of different size and five different statistical test with the aim of lowering the type I error [21]. The recommendation was, if it is feasible, to perform cross-validation of 5 runs of 2 folds, to obtain the lowest type I error. Otherwise, in case the size of the dataset only allows for one run, performs McNemar's test single train-test split and therefore there is no variance due to the repeated selection of train-test sets. Nadeau&Bengio identified the problem of high variance due to random selection of the train-test set that has not been considered in the previous work, and proposed a modified t-test to account for the randomness while still lowering type I and II errors [22]. Related to the cross-validation experiment design, Bouckaert et al. recommended to perform cross-validation repeating 10 times a partition of 10 fold for better repeatability of the statistical test [23].

Following these recommendations in this experiment has been used 10-folds times 10-repetitions cross-validation, in addition of using a global variable random_state of zero, which implies a random seed common to all experiments, to assure repeatability.

Significance Test

In addition to the model performance evaluation it is important to assess the statistical significance of the difference between models, to give validity to the empirical research, as it deals with data and its variability. There are two main approaches to statistical significance testing, the frequentist and Bayesian.

The frequentist significance testing approach is based on Hypothesis testing, namely the null and alternative hypotheses H0/Ha. The quality of the Hypotheses is evaluated by Type I and Type II errors, or False Positive and Negative as mentioned before. In this experiment the Null Hypothesis NullH is: does Model 1 perform better than Model 2?

To assess the type I and II error and therefore accept or reject the null hypothesis the frequentist approach uses the t-score, which has a traditional calculation and several revisions to adapt to the type of experiment. The modified t-test in conjunction with the Student's distribution is represented in **Equation 1** [22], where xj is the difference of the models performance at fold j, and σ is the estimate of the variance at the related fold.

$$t = \frac{\frac{1}{n}\sum_{j=1}^{n} x_j}{\sqrt{\left(\frac{1}{n}+\frac{n_{test}}{n_{train}}\right)\sigma^2}} \tag{1}$$

Nadeau&Bengio modification consisted on replacing the term $\frac{1}{n}$ in the denominator by $\left(\frac{1}{n}+\frac{n_{test}}{n_{train}}\right)$ representing the relationship between test/train sets at a particular fold or run.

The frequentist approach of this experiment is based on the Nadeau&Bengio modified t-test [22] and compared to the standard test, to corroborate the lack of consistency of the frequentist approach that heavily relies on the design of experiment, the way the NullH is posed and the statistical t-test used to prove the NullH.

The Bayesian approach to significance test is also denominated Bayesian t-test for correlated observations. Corani&Benavoli designed a Bayesian t-test to account for the dependency between samples due to the cross-validation [24]. The Bayesian likelihood function is modelled by the mean difference of the performance parameter and a noise vector. This noise vector has the form of a Multivariate Normal Distribution Noise with mean zero, and covariance proportional to the correlation matrix of the samples. Therefore, the covariance matrix takes into account of the correlation due to cross-validation. The prior distribution is a Normal-Gamma distribution is the conjugate for the likelihood function. The posterior marginal distribution over the mean difference of performance parameters is a Student's distribution.

Bayesian posterior distribution gives consistent information about: a) probability of equivalence between two classifiers, b) probability of one classifier being better than other, c) confidence intervals [25].

a) Probability of equivalence between two classifiers. By definition two classifiers are considered practically equivalent if they differ in less than 1% in their performance [26]. Therefore ROPE Region of Practical Equivalence is defined as the region of 1% within the two classifiers can be considered as equivalent. The Posterior distribution over the ROPE interval gives the region of equivalence between the two classifiers.

b) Probability of one classifier being better than other. The area under the posterior curve calculated in the interval from zero to infinity gives the probability of the performance of one classifier, compared with the area calculated in the interval from minus infinity to zero related to the second classifier.

c) Confidence intervals. The posterior distribution allows calculating the confidence intervals where the mean difference is found with 50%, 75% and 95% of probability.

3.1 Experiment Setup

The machine learning models were developed in Python 3.7.9 inside Kaggle Notebooks, a cloud computational environment based on Jupyter for code development. The localhost runs on Intel Core 7, graphic card Nvidia GeForce GT 635M, Windows 10, 64-bit. The Machine Learning Libraries for Python are Scikit-learn [19].

The dataset used was the network intrusion dataset NSL-KDD, a version with separate training and testing sets [27]. The training set has 125972 registers and 43 features and includes the target class for supervised learning, and the test set has 22543 registers and 43 data features. From the 43 data features were used four features: protocol type (3 categories), service (70 categories), flag (11 categories), attack type (23 categories).

Hyper-parameter variation

The cross-validation method can be used in combination with a parameter search method for selecting the parameters that give the best model performance. In this experiment two methods have been used, the exhaustive search, grid search, in the case of Decision Tree and Random Forest Classifiers, and randomized search in the case of Multilayer Perceptron Classifier. The use of randomized search instead of grid search was a compromise decision due to the time consuming of performing cross-validation combined with grid search, particularly for neural networks.

Decision Tree Hyper-parameter variation. In this article, four hyper-parameters were ranged for calibrating a Decision Tree Classifier: node splitting criterion, maximum number of features per tree, and minimum number of samples per leaf. The node splitting criterion can be selected between Gini and entropy measurement. The maximum number of features per tree can be selected between two methods sqrt(N), and log2(N), considering N the total number of features. The minimum number of samples per leaf is selected from a range between 10 and 50 in steps of 10. The maximum tree depth is selected from a range between 10 and 50 in steps of 10.

Random Forest Hyper-parameter variation. Three hyper-parameters were ranged for calibrating a Random Forest Classifier: number of estimators, and similarly to Decision Tree Classifier, maximum number of features per tree, and minimum number of samples per leaf. The number of estimators is drawn from a range between 10 and 50 in steps of 10. The maximum number of features per tree are selected between two methods sqrt(N), and log2(N), and the minimum number of samples per leaf is selected from a range between 10 and 50 in steps of 10.

Multilayer Perceptron Classifier Hyper-parameter variation. Four hyper-parameters were ranged for calibrating the Multi-Layer Perceptron Classifier: Size of hidden layer, activation function, alpha, and maximum number of iterations. Size of hidden layer was ranged between 50 and 200 in steps of 50. The activation function was selected from a set of Hyperbolic Tangent, Logistic Sigmoid Function and Rectifier Linear Unit. Alpha is the L2 Euclidean distance penalty and was selected from the set {0.0001, 0.001, 0.01, 0.1, 1, 10}. The maximum number of iterations range from 50 to 200 in steps of 50.

4 Result and Discussion

Different machine learning classifiers have been considered as candidates for NIDS. Classifiers such as Principal Component Analysis and Support Vector Machine were considered because the reported reliability in the classification performance. However, the trial training and testing runs did not get feasible results even with reduced set of features, for example using only the three features of protocol type, TCP, UDP and ICMP. Classifiers such as k-Nearest Neighbors kNN, Gaussian-Naïve Bayes GNB, and Logistic Regression LR neither of them gave good classification performance even performing a dedicated search in the classification algorithm.

The comparison of the accuracy distribution was useful as a preliminary analysis to understand the behaviour of the classifiers, before running a cross-validation method, which is time and resource demanding. **Figure 1 a)** shows the accuracy distribution for four classifiers, DTC, kNN, LR, and RFC, where the distribution of LR falls well below the rest of classifiers, misleading any possible comparison.

Once removed the classifiers that did not compare it was possible to get more precise values for better comparison. **Figure 1 b)** shows the performance distribution, ROC_AUC, for three classifiers, DTC, RFC, and MLPC, of comparable performance and candidates for cross-validation and hyper-parameter search.

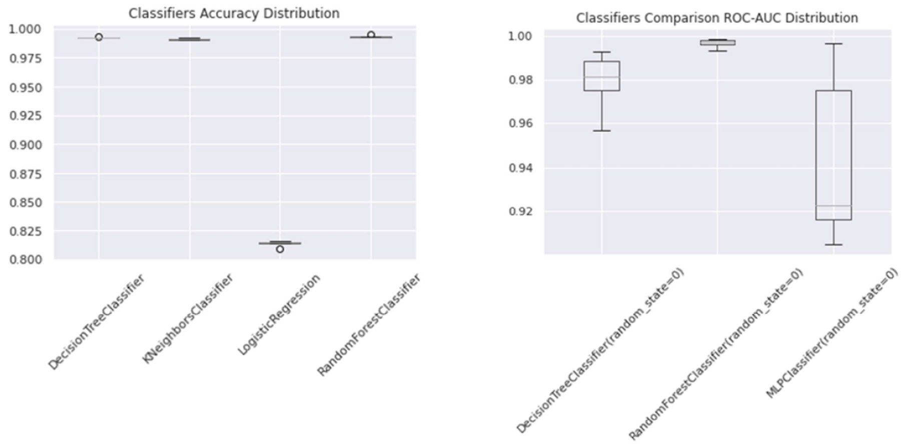


Figure 1. Performance Distribution for NIDS classifiers:
a) Accuracy Distribution for Decision Tree, K-Nearest-Neighbor, Logistic Regression, Random Forest Classifiers.
b) ROC_AUC Distribution for Decision Tree, Random Forest, and Multilayer Perceptron Classifiers.

As described in Section 3, the cross-validation method used is the 10-times repeated 10-folds, where the dataset is randomly rearranged and divided into 10 folds or sets out of which one set is used for testing and the remaining are used for training. This process repeats 10 times for better replicability, and the evaluation metrics are averaged across the runs to give an overall performance indicator.

The Repeated Stratified k-fold method implies that the scores resulting from the cross-validation are not independent, showing co-variation in the same folds because they are trained with the same set of data. **Figure 2** shows the cross-validation scores dependency from the test partition, where a great majority of scores present similar pattern per fold.

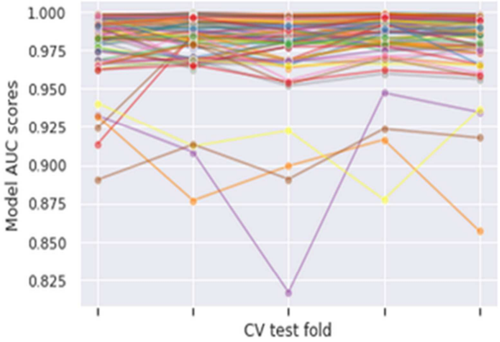
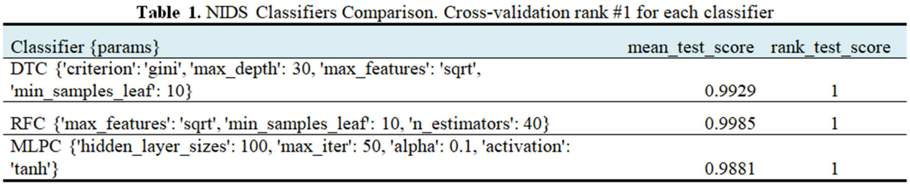


Figure 2. Cross-validation results dependency from the test partition: the scores present similar pattern per fold.

The result of cross-validation gave a ranking of classifiers for Decision Tree, Random Forest and Multilayer Perceptron. For the NIDS experiment the objective score was the ROC_Auc. **Table 1** shows the best classifiers, their parameters, and the mean test score.

Table 1. NIDS Classifiers Comparison. Cross-validation rank #1 for each classifier



Classifier {params}	mean_test_score	rank_test_score
DTC {'criterion': 'gini', 'max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 10}	0.9929	1
RFC {'max_features': 'sqrt', 'min_samples_leaf': 10, 'n_estimators': 40}	0.9985	1
MLPC {'hidden_layer_sizes': 100, 'max_iter': 50, 'alpha': 0.1, 'activation': 'tanh'}	0.9881	1

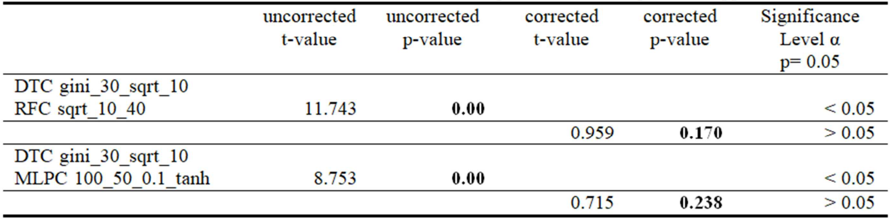
The similarity between classifiers performance shown in Table 1 is the result of a fine grained selection of models before running cross-validation. Firstly, classifiers with low performance were not considered for cross-validation, leaving out PCA, SVM, kNN, GNB and LR, as discussed in the beginning of Section 4. Therefore only DT, RFC, MLPC were selected as best candidates for comparing their performance. Several model variations are evaluated by the cross-validation method. The 10-times 10-fold cross-validation, leads to 100 folds to run. Additionally, these runs are combined with the search of the classifier hyper-parameters, resulting in 10000 fits for DTC, and 3200 fits for RFC, both with exhaustive search, and 1000 fits for MLP with randomized search. That means high competitive models with similar scores. However, it is needed further evaluation of the quality of the experiments following statistical analysis to assess the validity of the results.

Statistical significance test: Frequentist approach

The statistical significance of the classifiers comparison, according to the frequentist approach is in **Table 2.** The values are the corrected t- and p-values according to Nadeau&Bengio’s t-test compared to the uncorrected ones.

From the results in the Table 2 there is no conclusive acceptance of the Null Hypothesis NullH “the first model performs better than the second”. Considering a significance level of 95%, a p-value higher than 0.05 indicates acceptance of the NullH, the corrected p-value indicates acceptance of the NullH, contrary to the uncorrected value. This controversy has been mentioned in the literature criticizing the frequentist inference method and suggesting the Bayesian statistical inference as a more rigorous approach [28].

Table 2. Significance Test, Frequentist approach.
T-test results, uncorrected and corrected according to Nadeau&Bengio t-test approach



	uncorrected t-value	uncorrected p-value	corrected t-value	corrected p-value	Significance Level α p= 0.05
DTC gini_30_sqrt_10 RFC sqrt_10_40	11.743	**0.00**			< 0.05
			0.959	**0.170**	> 0.05
DTC gini_30_sqrt_10 MLPC 100_50_0.1_tanh	8.753	**0.00**			< 0.05
			0.715	**0.238**	> 0.05

Statistical significance test: Bayesian approach

Bayesian posterior distribution over the mean difference of the classifier performance is described in **Table 3**.

a) Equivalence in Performance. The posterior distribution over the ROPE interval of +/-1% (-0.01, 0.01) gives the region of practically equivalence in their performance between the two classifiers. The results give the probability of 0.769 for the DTC and RFC classifiers of being practically equivalent, and the probability of 0.803 for the DTC and MLPC. That means, DTC and MLPC have higher probability of having equivalent performance sharing higher area under the curve. The particular model configurations are as follows: for DTC, the node splitting criterion is Gini, the minimum number of samples per leaf is 30, the maximum number of features per tree is sqrt(N), where N is the total number of features, and the maximum tree depth is 10; and for MLPC, the size of hidden layer is 100, the maximum number of iterations is 50, the L2 Euclidean distance Alpha is 0.1, and the activation function is the Hyperbolic Tangent (tanh).

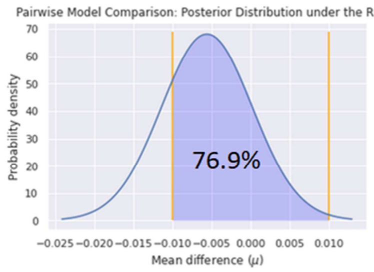
b) Performance comparison. Comparing DTC and RFC, the probability of DTC classifier being better than RFC is 0.769, which is outperformed by the probability of RFC being better than DTC, which is 0.830. Comparing DTC and MLPC, the probability of DTC being better than MLPC is 0.803, which is higher than the probability of MLPC of being better than DTC, which is 0.238. Concluding that RFC outperforms the others classifiers in accordance with the ranked scores.

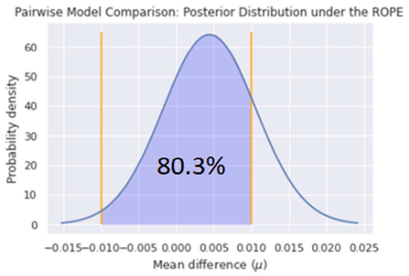
c) Confidence intervals. The Table 3 c) shows the confidence intervals, where the mean difference is found, with 50%, 75% and 95% of probability respectively.

Table 3. Significance Test, Bayesian Approach.
Bayesian Posterior distribution over the mean difference of the models performance.

Comparison 1 Model 1: DTC gini_30_sqrt_10 Model 2: RFC sqrt_10_40	**Comparison 2** Model 1: DTC gini_30_sqrt_10 Model 2: MLPC 100_50_0.1_tanh

a) Performance Equivalence. ROPE Region of Practical Equivalence





Probability of model 1 and model 2 being practically equivalent: **0.769**	Probability of model 1 and model 2 being practically equivalent: **0.803**

b) Performance Comparison

Probability of model 1 better performance than model 2: **0.170** Probability of model 2 being more accurate than model 1: **0.830**	Probability of model 1 better performance than model 2: **0.762** Probability of model 2 better performance than model 1: **0.238**

c) Confidence Intervals: 50%, 75% and 95%

interval	lower value	upper value
0.50	**-0.009557**	**-0.001647**
0.75	**-0.012362**	**0.001159**
0.95	**-0.017194**	**0.005991**

interval	lower value	upper value
0.50	**0.000234**	**0.008632**
0.75	**-0.002745**	**0.011610**
0.95	**-0.007875**	**0.016740**

Classifiers Performance Results

The classifiers performance is evaluated by the confusion matrix, ROC curve, and classification report. **Tables 4, 5 and 6** summarize the performance of the best classifiers selected by cross-validation.

The **confusion matrix** gives the relation between the actual and predicted target class, resulting in True Positive-Negative TP-TN, and False Positive-Negative FP-FN values. The best performance is higher TP and TN cases, which are lighter colors in the heat map. Visually the three classifiers have similar performance, lighter colors in the diagonal for TP and TN cases. However, RFC shows the highest numbers of TP and TN, and lowest numbers of FP and FN.

The **Receiver Operating Characteristic ROC** curve plots the rate of TPR vs FPR at different classification thresholds, and the Area under the ROC Curve AUC score is the aggregate value which can be calculated using the trapezoidal rule or using the trial scores. The trapezoidal rule has been used in this experiment for better accuracy of the indicator. ROC curves ratify the confusion matrix results, adding description of the relationship between TP and FP. The ROC curve for RFC has sharp response throughout the range, meaning higher TPR and lowest FPR, while the ROC curves for DT and MLPC show and increment of FPR to the expense of TPR.

Table 4. Performance results of the best NIDS classifiers: Decision Tree Classifier

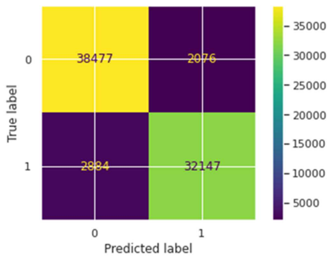
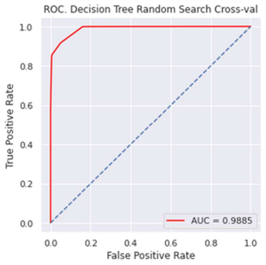
	Confusion Matrix	ROC curve
DTC Hyper Parameter Variation criterion: gini, entropy max_depth: range(10,50,10) max features: sqrt, log2 min_samples_leaf: range(10.50,10)		

Table 5. Performance results of the best NIDS classifiers: Random Forest Classifier

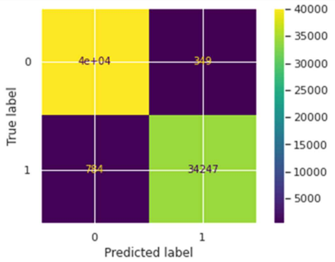
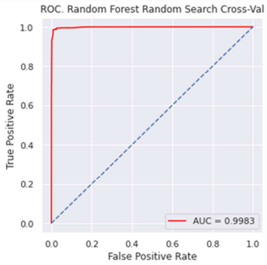
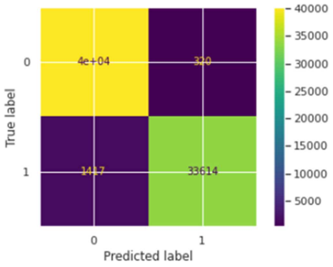
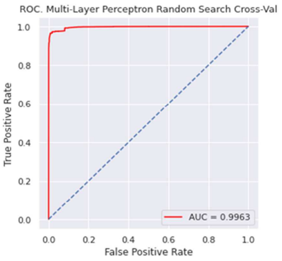
	Confusion Matrix	ROC curve
RFC Hyper Parameter Variation max features: sqrt, log2 min_samples_leaf: range(10.50,10) n_estimators: range(10,50,10)		

Table 6. Performance results of the best NIDS classifiers:
Multilayer Perceptron Classifier

	Confusion Matrix	ROC curve
MLPC Hyper Parameter Variation activation: tanh, logistic, relu alpha: [0.0001,0.001,0.01,0.1,1,10] hidden_layer_size: range(50,200,50) max_iter: range(50,200,50)		

The Classification report in **Table 7** summarizes precision, recall, f1-score, and the supporting sample for the two target classes, normal flow and attack. Defining Precision as the relationship between TP/(TP+FP), which means how many selected cases are relevant; and Recall as TP/(TP+FN), which represents how many relevant cases are selected. The score F1 is defined as the weighted average of the precision and recall, with best value towards one [19]. The results in Table 7 show highest Precision, Recall, and F1-score for RFC to detect normal flow and attacks compared to DTC and MLPC. Despite the MLPC performance is closer to RFC, the weighted average of Precision is outperformed by RFC with 0.99.

Table 7. Classification Report of the best NIDS classifiers

		DTC			RFC			MLPC		
	Support	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
normal	40553	0.93	0.95	0.94	0.98	0.99	0.99	0.97	0.99	0.98
attack	35031	0.94	0.92	0.93	0.99	0.98	0.98	0.99	0.96	0.97
accuracy	75584			0.93			0.99			0.98
macro avg	75584	0.93	0.93	0.93	0.99	0.98	0.98	0.98	0.98	0.98
weighted avg	75584	0.93	0.93	0.93	0.99	0.99	0.99	0.98	0.98	0.98

5 Conclusions

The ML classifiers algorithms considered in this work for Network Intrusion Detection were Principal Component Analysis, Support Vector Machine, k-Nearest Neighbor, Gaussian-Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Multilayer Perceptron, to classify 4 features of the network flow, totalizing 107 categories. From those, only DTC, RFC and MLPC gave better performance.

To assess the classifier performance and select the best candidate, it has been explored the applicability of the cross-correlation method combined with hyper-parameter search methods to further refine the range of available model options. However, cross-correlation becomes expensive as the dataset and number of parameters increases. To overcome this drawback, randomized search has been used in the hyper-parameter search for MLPC, which is the most expensive algorithm due to the complexity of network structure; preferring exhaustive search for DTC and RFC. As a result the experiment consisted of 14200 fits (10000 fits for DTC, 3200 fits for RFC, and 1000 fits for MLP), which means a high refined set of models, to give high performance candidates with competitive results.

The similarity in the performance scores of the models leads to the use of statistical validation methods to assess the quality of the experiments. The analysis by Bayesian Posterior Distribution has been proved to be rigorous and consistent method for statistical significance test, supporting the selection of the RFC as the classifier with better performance in this particular case of classifiers in Network Intrusion Detection.

6 Bibliography

[1] M. Roesch, “Snort - Lightweight Intrusion Detection for Networks,” in *Proceedings of the 13th USENIX Conference on System Administration*, USA, 1999, pp. 229–238.

[2] K. Nam and K. Kim, “A Study on SDN security enhancement using open source IDS/IPS Suricata,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2018, pp. 1124–1126. doi: 10.1109/ICTC.2018.8539455.

[3] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, “A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection,” *IEEE Commun. Surv. Tutor.*, vol. 21, no. 1, pp. 686–728, Firstquarter 2019, doi: 10.1109/COMST.2018.2847722.

[4] Y. Gu, K. Li, Z. Guo, and Y. Wang, “Semi-Supervised K-Means DDoS Detection Method Using Hybrid Feature Selection Algorithm,” *IEEE Access*, vol. 7, pp. 64351–64365, 2019, doi: 10.1109/ACCESS.2019.2917532.

[5] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, “Autoencoder-based feature learning for cyber security applications,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3854–3861. doi: 10.1109/IJCNN.2017.7966342.

[6] M. Z. Alom and T. M. Taha, “Network intrusion detection for cyber security using unsupervised deep learning approaches,” in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, Jun. 2017, pp. 63–69. doi: 10.1109/NAECON.2017.8268746.

[7] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, “Predicting network attack patterns in SDN using machine learning approach,” in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2016, pp. 167–172. doi: 10.1109/NFV-SDN.2016.7919493.

[8] M. Latah and L. Toker, “Towards an efficient anomaly-based intrusion detection for software-defined networks,” *IET Netw.*, vol. 7, no. 6, pp. 453–459, 2018, doi: 10.1049/iet-net.2018.5080.

[9] A. Dawoud, S. Shahristani, and C. Raun, “A Deep Learning Framework to Enhance Software Defined Networks Security,” in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, May 2018, pp. 709–714. doi: 10.1109/WAINA.2018.00172.

[10] D. Mudzingwa and R. Agrawal, “A study of methodologies used in intrusion detection and prevention systems (IDPS),” in *2012 Proceedings of IEEE Southeastcon*, Mar. 2012, pp. 1–6. doi: 10.1109/SECon.2012.6197080.

[11] I. Homoliak, M. Teknös, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, “Improving Network Intrusion Detection Classifiers by Non-payload-Based Exploit-Independent Obfuscations: An Adversarial Approach,” *ICST Trans. Secur. Saf.*, vol. 5, p. 156245, Jan. 2019, doi: 10.4108/eai.10-1-2019.156245.

[12] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, “Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection,” *IEEE Access*, vol. 6, pp. 33789–33795, 2018, doi: 10.1109/ACCESS.2018.2841987.

[13] A. Ahmad, E. Harjula, M. Ylianttila, and I. Ahmad, *Evaluation of Machine Learning Techniques for Security in SDN*. 2020.

[14] Scikit-learn, “Statistical comparison of models using grid search — scikit-learn 0.24.2 documentation.” https://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_stats.html#id12 (accessed Jun. 22, 2021).

[15] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, “The Elements of Statistical Learning: Data Mining, Inference, and Prediction,” *Math Intell*, vol. 27, pp. 83–85, Nov. 2004, doi: 10.1007/BF02985802.

[16] A. Hershy, “Gini Index vs Information Entropy,” *Medium*, Oct. 15, 2020. https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb (accessed Apr. 08, 2021).

[17] J. Taylor and R. J. Tibshirani, “Statistical learning and selective inference,” *Proc. Natl. Acad. Sci.*, vol. 112, no. 25, pp. 7629–7634, Jun. 2015, doi: 10.1073/pnas.1507583112.

[18] S. Kotsiantis, “Bagging and boosting variants for handling classifications problems: A survey,” *Knowl. Eng. Rev.*, vol. 29, pp. 78–100, Jan. 2013, doi: 10.1017/S0269888913000313.

[19] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[20] R. Collobert and S. Bengio, “Links between perceptrons, MLPs and SVMs,” in *Proceedings of the twenty-first international conference on Machine learning*, New York, NY, USA, Jul. 2004, p. 23. doi: 10.1145/1015330.1015415.

[21] T. G. Dietterich, “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms,” *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998, doi: 10.1162/089976698300017197.

[22] C. Nadeau and Y. Bengio, “Inference for the Generalization Error,” *Mach. Learn.*, vol. 52, no. 3, pp. 239–281, Sep. 2003, doi: 10.1023/A:1024068626366.

[23] R. R. Bouckaert and E. Frank, “Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms,” in *Advances in Knowledge Discovery and Data Mining*, Berlin, Heidelberg, 2004, pp. 3–12. doi: 10.1007/978-3-540-24775-3_3.

[24] G. Corani and A. Benavoli, “A Bayesian approach for comparing cross-validated algorithms on multiple data sets,” *Mach. Learn.*, vol. 100, no. 2, pp. 285–304, Sep. 2015, doi: 10.1007/s10994-015-5486-z.

[25] S. Shikano, “Hypothesis Testing in the Bayesian Framework,” *Swiss Polit. Sci. Rev.*, vol. 25, no. 3, pp. 288–299, 2019, doi: https://doi.org/10.1111/spsr.12375.

[26] J. K. Kruschke, “Rejecting or Accepting Parameter Values in Bayesian Estimation,” *Adv. Methods Pract. Psychol. Sci.*, vol. 1, no. 2, pp. 270–280, Jun. 2018, doi: 10.1177/2515245918771304.

[27] “NSL-KDD Datasets, Research, Canadian Institute for Cybersecurity, University of New Brunswick.” https://www.unb.ca/cic/datasets/nsl.html (accessed Apr. 07, 2021).

[28] A. Benavoli, G. Corani, J. Demsar, and M. Zaffalon, “Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis,” *J. Mach. Learn. Res.*, vol. 18, Jun. 2016.