Network Intrusion Detection with XGBoost

Arnaldo Gouveia[1,2] and Miguel Correia[2]

[1] IBM Belgium – Brussels, Belgium
[2] INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Lisboa, Portugal

Abstract. XGBoost is a recent machine learning method that has been getting increasing attention. It won Kaggle's Higgs Machine Learning Challenge, among several other Kaggle competitions, due to its performance. In this , we explore the use of XGBoost in the context of anomaly-based network intrusion detection, an area in which there is a considerable gap. We study not only the performance of XGBoost with two recent datasets, but also how to optimize its performance and model parameter choice. We also provide insights into which dataset features are best for performance tuning.

1 Introduction

The detection of security-related events using *Machine Learning* (ML) has been extensively investigated [9–11, 14, 18, 20, 22, 26, 29]. *Intrusion Detection Systems* (IDSs) are security tools used to detect malicious activity. *Network Intrusion Detection Systems* (NIDS) are one of the best known contexts of machine learning application in the security field [14]. IDSs can be classified using several criteria [9]. One of these criteria is the detection approach, in terms of which IDSs (and NIDSs) can be signature-based or anomaly-based. The former class detects attacks by comparing the data flow under analysis to patterns stored in a signature database of known attacks. The later detects anomalies using a model of normal behaviour of the monitored system and flagging behavior lying outside of the model as anomalous or suspicious. Signature-based IDSs can detect well-known attacks with high accuracy but fail to detect or find unknown attacks [18, 29], whereas anomaly-based IDSs have that capacity. In this we focus on using ML classifiers for detection, which is a form of signature-based detection.

A sophisticated attacker can bypass these techniques, so the need for more intelligent intrusion detection is increasing by the day. Researchers are attempting to apply ML techniques to this area of cybersecurity. Supervised machine learning algorithms, when applied to historical alert data, can significantly improve classification accuracy and decrease research time for analysts. It can supplement analysts with additional data and insights to make better judgment calls. Though prediction models based on historical data can improve analyst productivity, they will never replace security analysts altogether.

NIDS serve as one of the critical components for a defense-in-depth solution. NIDS appliances allow for active, inline protection for known and unknown threats passing across a network segment at all layers of the OSI model. The employment, tuning, and upkeep of machine learning models on a NIDS may lead to a negative impact on production traffic if not properly maintained. This document serves as guidance to help shape the development of an NIDS systems machine learning model management approach. Through proper maintenance, placement, and tuning of models, an unwanted impact to network traffic can be kept to a minimum while also achieving an optimal balance of security and network performance.

XGBoost (EXtreme Gradient Boosting) [7] is a recent decision-tree-based ensemble machine learning algorithm. XGBoost won Kaggle's Higgs Machine Learning Challenge in 2014, that consisted in searching for Higgs bosons [16] in a large dataset provided by CERN[3] [1,2]. The dataset and the subject of the challenge correspond to a particular physical phenomena, the Higgs boson to *Tau* particles decay ($H \rightarrow \tau^{+}\tau^{-}$) [28]. The Higgs boson has many different processes (called channels by physicists) through which it can decay, i.e., produce other particles. The Standard Model of particle physics has been being developed for decades and the existence of almost all subatomic particles has been confirmed experimentally in the past century. The Higgs boson was the exception, as it was detected experimentally only on 2012, after experiments in the Large Hadron Collider (LHC) [27,28]. XGBoost found again Higgs bosons on CERN data only 2 years later.

XGBoost is gaining popularity due to its performance and scalability [2,7,23]. XGBoost has been proving faster than other popular algorithms in a single machine and to scale to billions of examples in distributed or memory-limited settings as shown empirically in Kaggle competitions [23]. Kaggle is an online community focused on machine learning, popular for the competitions it organizes.

The goal of a NIDS is to generate alerts when adversaries try to penetrate or attack the network. Consider a *flow* to be a sequence of IP packets with similar features (e.g., same destination IP address and port). Typically an intrusion involves a few flows hidden among many legitimate flows. In statistical terms, the problem of detecting a few flows in a large set of flows is similar to the problem of detecting Higgs bosons. In this scope we aim at answering the following questions:

1. How effectively can XGBoost be used in the context of NIDSs?
2. How can machine learning model management be optimized in NIDS?
3. What insights can we devise in this approach?

We use XGBoost with two datasets carefully designed for evaluating NIDSs machine learning algorithms: UNB NSL KDD [26] and UNSW NB15 [20,22]. XGBoost has been barely used for intrusion detection, as explained in the next section.

[3] Conseil Européen pour la Recherche Nucléaire.

The contributions of the this work are: (1) a study of the use of XGBoost in the context of network intrusion detection; (2) an optimization method for XGBoost for achieving performance in that context; (3) a set of XGBoost parameters with strong impact/correlation on performance. We used the H2O XGBoost implementation, written in R, in the experiments. H2O is an open source software package machine learning.

2 Related Work

There are a few works related to ours in the sense that they use XGBoost for intrusion detection. However, they are very limited, e.g., because they consider only the detection of DDoS attacks or focus on the specific case of Software Defined Networks (SDN).

Chen at al. [8] focused on applying XGBoost to learn to identify a DDoS attack in the controller of an SDN cloud. They used the old 1999 ACM KDD Cup dataset[4]. The authors have shown that XGboost can detect DDoS attacks by analyzing attack traffic patterns. The XGBoost algorithm has shown higher accuracy and lower false positive rate than other algorithms tested (Gradient Boosting Decision Tree, Random Forest and Support Vector Machines), according to the authors. In addition, XGBoost proved to be quicker than the other algorithms tested.

Bansal and Kaur [5] used the CICIDS 2017 dataset again to detect DDoS attacks, not other classes [24]. XGBoost proved again to provide better accuracy values than KNN, AdaBoost, MLP and Naïve-Bayes.

In a short paper, Amaral et al. [4] used XGBoost to classify malicious traffic in SDNs. The authors present a simple architecture deployed in an enterprise network that gathers traffic data using the OpenFlow protocol and show how several machine learning techniques can be applied for traffic classification, including XGBoost. Traffic of a number of applications (e.g., BitTorrent, YouTube) was classified and the comparative results of XGboost in face of other classifiers (Random Forests and Stochastic Gradient Boosting) proved comparable or better.

Dhaliwal et al. also used XGBoost for detection [10]. However, they present essentially an exercise of running XGBoost on a single dataset that does not cover the second and third questions above.

Mitchell et al. [19] presented a CUDA based implementation of a decision tree construction algorithm by using XGBoost. The algorithm has been executed entirely on the GPU and has shown high performance with a variety of datasets and settings, including sparse input matrices.

[4] `https://www.kdd.org/kdd-cup/view/kdd-cup-1999`

3 XGBoost and Boosting Trees

XGBoost is based on a set of machine learning concepts:

- *Weak learners:* XGBoost uses *decision trees* as *weak* learners;
- *Boosting:* combines the contributions of many weak learners to produce a strong learner;
- *Gradient boosting:* the notion of doing boosting by minimizing errors by introducing a gradient term.

Three main forms of gradient boosting are supported:

- Gradient Boosting algorithm also called gradient boosting machine including the learning rate;
- Stochastic Gradient Boosting with sub-sampling at the row, column and column per split levels;
- Regularized Gradient Boosting with both L1 and L2 regularization.

3.1 XGBoost general formalism and tuning parameters

The model concept in supervised learning (and in machine learning in general) refers to the mathematical processing by which the prediction y_i are made from the input x_i.

The model parameters are the undetermined factors that to learn from data in order to have a model that can make good predictions. The model optimization is executed by means of a function to be maximized or minimized in order to find out such parameters – the objective function.

Modelling XGBoost. An important characteristic of objective functions is that they consist of two parts, training loss and the regularization term:

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta) \tag{1}$$

where L is the training loss function, and Ω is the regularization term. The training loss measures the capacity of the model to predict data that is also in the training dataset. A common choice of L is the mean squared error, which is given by $L(\theta) = \sum_i (y_i - \hat{y}_i)^2$.

Another commonly used loss function is logistic loss, to be used for logistic regression:

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})] \tag{2}$$

Mathematically, we can write the XGboost model in the form

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{3}$$

where K is the number of trees, f is a function in the functional space $s\mathcal{F}$ set of all possible classification and regression trees (CARTs). The objective function to be optimized is given by

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4}$$

The tree parameters to be used are those that can be determined by functions f_i each containing the structure of the tree and the leaf scores. It is considered intractable to learn all the trees at once. Instead, an additive strategy is used: iterate adding one new tree at a time. The prediction value at step t as $\hat{y}_i^{(t)}$. Then we have:

$$\begin{aligned} \hat{y}_i^{(0)} &= 0 \\ \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ &\dots \\ \hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \end{aligned} \tag{5}$$

In each step a tree is added that optimizes the objective function.

$$\begin{aligned} \text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{constant} \end{aligned} \tag{6}$$

If we consider using mean squared error (MSE), as in the Ridge approach, as our loss function, the objective becomes

$$
\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n}(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n}[2(\hat{y}_i^{(t-1)} - y_i)f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + \text{constant}
\end{aligned}
\tag{7}
$$

With the Taylor expansion of the loss function up to the second order we obtain:

$$
\text{obj}^{(t)} = \sum_{i=1}^{n}[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) + \text{constant} \tag{8}
$$

where the g_i and h_i are defined as

$$
\begin{aligned}
g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \\
h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})
\end{aligned}
\tag{9}
$$

After we remove all the constants, the specific objective at step t becomes

$$
\sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) \tag{10}
$$

This becomes our optimization goal for the new tree. One important advantage of this definition is that the value of the objective function depends only on the terms g_i and h_i. This is the reason why XGBoost allows using different custom loss functions. We can optimize every loss function, including logistic regression and pairwise ranking, using exactly the same solver that takes g_i and h_i as input.

Model Complexity. To determine the complexity of the tree $\Omega(f)$ the definition of the tree $f(x)$ can be defined as:

$$
f_t(x) = w_{q(x)}, w \in R^T, q : R^d \to \{1, 2, \cdots, T\} \tag{11}
$$

where T is the number of leaves and w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf.

A commonly accepted definition of complexity in XGboost is given by:[5]

[5] https://xgboost.readthedocs.io/en/latest/tutorials/model.html

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{12}$$

The structure score. After re-formulating the tree model, we can write the objective value with the t-th tree as:

$$\begin{aligned} \text{obj}^{(t)} &\approx \sum_{i=1}^{n}[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \end{aligned} \tag{13}$$

The term $I_j = \{i|q(x_i) = j\}$ refers to the set of indices of data points assigned to the j-th leaf. In the second line the index of the summation changed because all the data points on the same leaf get the same score. Defining $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$ the expression becomes:

$$H_j = \sum_{i\in I_j} h_i \tag{14}$$

The terms w_j are independent with respect to each other, the form $G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2$ is quadratic and the best w_j for a given tree structure $q(x)$ and the best objective reduction we can get is:

$$\begin{aligned} w_j^* &= -\frac{G_j}{H_j + \lambda} \\ \text{obj}^* &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \end{aligned} \tag{15}$$

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \tag{16}$$

Further insight into regularization approaches. To address overfitting, i.e., to avoid modelling the training data too tightly, XGBoost in the `H2O` R Package [3] uses L1 and L2 *regularization*. The difference between these two penalty terms is mainly:

– *Lasso Regression (L1):* adds the modulus of the coefficient vector as penalty term to the loss function.

– *Ridge regression (L2):* adds a squared modulus term of the coefficient vector as penalty term to the loss function.

The key difference between these techniques is that Lasso shrinks the less important features' coefficients to zero, thus effectively removing some features, something that Ridge does not manage to do. Feature reduction is, therefore, a more defined consequence of using Lasso. XGBoost's regularization strategy involves also configuring other parameters to be effective, e.g., *Maximum Depth*, *Minimum Child Weight*, and *Gamma*. Without proper regularization, the tree may split until it can predict the training set perfectly, i.e., may overfit. Therefore, it will end up losing generality and will not perform as good on new data (i.e., on test data or on production data).

The trees used play the role of weak learners, providing decisions only slightly better than a random guess. XGBoost used an ensemble approach, i.e., it combines a large number of decision trees to produce a final model consisting of a forest of decision trees. An individual decision tree is grown by ordering all features and checking each possible split for every feature. The split that results in the best score for the chosen objective function becomes the rule for that node.

For a given data set with n examples and m features $\mathcal{D} = \{(x_i, y_i)\}$ ($|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$), a tree ensemble model uses K additive functions to predict the output [7]:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}, \tag{17}$$

where $\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ is the space of regression trees. In the Equation, q represents the structure of each tree that maps an example to the corresponding leaf index, T is the number of leaves in the tree, each f_k corresponds to an independent tree structure q, and w are leaf weights.

In the general case as stated before an objective functions has two terms: training loss and regularization [7]: $\mathcal{L}(\theta) = L(\theta) + \Omega(\theta)$. A common choice of $L(\theta)$ is the mean squared error. With proper choices for y_i we can express regression, classification, and ranking. Training the model amounts to finding the parameters θ that best fit the training data x_i labels y_i. In order to train the model, we need to define the objective function to measure how well the model fits the training data. To learn the set of functions used in the model, we minimize the following regularized objective:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{18}$$

Table 4.1. Dataset comparison. The UNB NSL KDD files used were the *KDDTest+.arff* and the *KDDTrain+.arff* dataset files.

Year	Dataset	Size	Types of attack

2016	NSL-KDD	Train= 18.7MB
		Test= 3.4MB
			Probing, DoS, R2L, U2R

2017	UNSW NB15	Train= 14.7MB
		Test= 30.8MB
			Fuzzers, Shellcode, Analysis, Backdoors, DoS,
			Exploits, Generic, Reconnaissance, Shellcode, Worms

where l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i.

The second term Ω penalizes the complexity of the model (i.e., the regression tree functions). Depending on the type of regularization chosen, $\Omega(f)$ is usually defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\| \text{ (L1 or Lasso) or} \tag{19}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2 \text{ (L2 or Ridge)} \tag{20}$$

where w is the vector of scores on leaves, T is the number of leaves, γ and λ are adjustable gain factors. This regularization term helps to avoid overfitting.

4 The Datasets

An important component of an effective NIDS evaluation is a good benchmark dataset. We use two recent datasets that have been designed specifically and carefully to evaluate NIDSs. A comparative summary of the two can be found in Table 4.1.

4.1 The UNB NSL KDD dataset

The University of New Brunswick (UNB) NSL KDD intrusion detection evaluation dataset was created using a principled approach, that involved defining features that maximize three costs [25]. The data are sequences of entries labeled as *normal* or *attack*. Each entry contains a set of characteristics of a *flow*, i.e., of a sequence of IP packets starting at a time instant and ending at another, between which data flows between two IP addresses using a transport protocol (TCP, UDP) and an application-layer protocol (HTTP, SMTP, SSH, IMAP, POP3, or FTP). The dataset is fairly balanced with prior class probabilities of 0.465736 for the *normal* class and 0.534264 for the *anomaly* class. The attacks represented in dataset fall in four classes:

Table 4.2. Attacks in the UNB NSL KDD train and test datasets (all attacks from the first exist also in the second).

Class	Train dataset attacks	Test dataset only attacks
Probing	portsweep, ipsweep, satan, guesspasswd, spy, nmap	snmpguess, saint, mscan, xsnoop
Denial of Service (DoS)	back, smurf, neptune, land, pod, teardrop, buffer overflow, warezclient, warezmaster	apache2, worm, udpstorm, xterm
Remote to Login (R2L)	imap, phf, multihop	snmpget, httptunnel, xlock, sendmail, ps
User to Root (U2R)	loadmodule, ftp write, rootkit	sqlattack, mailbomb, processtable, perl

The UNB NSL KDD dataset is composed of two sub-datasets: a *train dataset*, used for training a NIDS, and a *test dataset*, used for testing. Both have the same structure and contain all four types of attacks. However, the test dataset has more attacks as shown in Table 4.2, to allow evaluating the ability of algorithms to generalize. Each record of the dataset is characterized by features that fall into three categories: basic, content related, and traffic related. These features are represented in Table 4.3.

4.2 The UNSW NB15 dataset

The second dataset comes from the Australian Centre for Cyber Security (ACCS) [20, 22]. The IXIA PerfectStorm tool[6], an application traffic simulator, has been used in the ACCS Cyber Range Lab to create a dataset with normal traffic and abnormal synthetic network traffic. The abnormal traffic is created with the IXIA tool, which contains information about new attacks updated using MITRE's Common Vulnerabilities and Exposures (CVE) site. The number of records of the training set is $175,341$; the testing set has $82,332$ records from the different types, attack and normal. The UNSW NB15 data set includes nine categories of attacks:

- *Fuzzers*: in this type of attack, randomly generated data is feed into a suspended program or network;
- *Reconnaissance*: The attacker gathers information about a system for later attacking it;
- *Shellcode*: code is used as the payload in an attack packet;
- *Analysis*: includes port scan, spam and HTML files penetrations;
- *Backdoors*: access to a system is gained by bypassing any secured layers;
- *Denial of Service*: the perpetrator intents resource unavailability by temporarily or indefinitely disrupting services;
- *Exploits*: the attacker exploits vulnerabilities of the system through known loopholes of the system;

[6] `https://www.ixiacom.com/products/perfectstorm`

Table 4.3. Features used to characterize each flow in the UNB NSL KDD dataset: basic (top), content (middle), traffic (bottom).

Feature	Description
duration	length of the flow in seconds
protocol-type	type of protocol, e.g., TCP, UDP, ICMP
service	network service, e.g., HTTP, Telnet
src-bytes	num. of data bytes from source to destination
dst-bytes	num. of data bytes from destination to source
flag	status of the flow, normal or error
lang	1 if flow is for the same host/port; 0 otherwise
wrong-fragment	num. of erroneous fragments
urgent	num. of urgent packets
hot	num. of hot indicators
num-failed-logins	num. of failed login attempts
logged-in	1 if successfully logged in; 0 otherwise
num-compromised	num. of compromised conditions
root-shell	1 if root shell is obtained; 0 otherwise
su-attempted	1 if su root command attempted; 0 otherwise
num-root	num. of root accesses
num-file-creations	num. of file creation operations
num-shells	num. of shell prompt
num-access-files	num. of operations on access control files
num-outbound-cmds	num. of outbound commands in a ftp session
is-host-login	1 if the login belongs to the hot list; 0 otherwise
is-guest-login	1 if the login is a guest login; 0 otherwise
count	num. of connections to the same host as current
serror-rate	% of connections that have SYN errors
rerror-rate	% of connections that have REJ errors
same-srv-rate	% of connections to the same service
diff-srv-rate	% of connections to different services
srv-count	num. of connections to the same service as current
srv-serror-rate	% of connections that have SYN errors
srv-rerror-rate	% of connections that have REJ errors
srv-diff-host-rate	% of connections to different hosts
dst-host-count	num. of connections to the same destination host
dst-host-srv-count	num. of connections to the same service as current
dst-host-same-srv-rate	% of connections to the same service
dst-host-diff-srv-rate	% of connections to different services
dst-host-same-src-port-rate	% of connections from same source and port
dst-host-srv-diff-host-rate	% of connections to different services
dst-host-serror-rate	% of connections that have SYN errors
dst-host-srv-serror-rate	% of connections that have SYN errors per service
dst-host-rerror-rate	% of connections that have REJ errors
dst-host-srv-rerror-rate	% of connections that have REJ errors per service

– *Generic*: the attack is implemented without knowing how the cryptographic primitive is implemented and works for all block ciphers;
– *Worms*: the attack mechanism replicates itself through the network.

The UNSW NB15 dataset includes 9 different moderns attack types (compared to 21 older attack types in UNB NSL KDD dataset) and wide varieties of real normal activities as well as 49 features inclusive of the class label consisting total of $2,540,044$ records. These features are categorized into six groups of features: flow related, Basic Features, content related, time related, additional generated and labelled features. These features include a variety of packet-based features and flow related features. The packet based features assist the examination of the payload beside the headers of the packets. On the contrary, for the flow based features and maintaining low computational analysis instead of observing all the packets going through a network link, only connected packets of the network traffic are considered. Moreover, the flow related features are based on traffic direction, inter-arrival time and inter-packet length. The matched features are categorised into several groups (Table 4.4).

5 Performance Metrics

We have chosen *logloss* as the primary criteria for assessing the model's performance, and used the *Area Under Curve* (AUC) to obtain additional insights. Moreover, we show that there are a number of other metrics that are closely correlated to those, so optimizing one will likely provide comparable results for others (Section 7.2).

logloss (logarithmic loss) is a metric used to evaluate the performance of both binomial and multinomial classifiers [17]. logloss evaluates how close the values predicted by a model (uncalibrated probability estimates) are to the actual target value. In other words this metric captures the extent to which predicted probabilities diverge from class labels: it incentives well-calibrated probabilities. The metric is shown in Equation 21. In the equation, N is the total number of rows (observations) of the dataset; w is user-defined weight per-row (defaults is 1); p is the predicted value (uncalibrated probability) assigned to a given row by the algorithm; y is the actual target value.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} w_i(\ y_i \ln(p_i) + (1 - y_i)\ln(1 - p_i)\ ) \tag{21}$$

Receiver operating characteristic (ROC) graphs (or curves) are useful for visualizing the performance of classifiers [13]. ROC graphs have, in recent years, been increasingly used in machine learning and data mining research. The AUC, i.e., the area under a ROC curve, represents the probability that a given sample classification is (correctly) rated or ranked with greater suspicion than a randomly chosen classification [6,15].

The ROC curve is a useful tool for a few reasons: The curves of different models can be compared directly in general or for different thresholds. Also the AUC can be used as a summary of the model skill. The shape of the curve is informative in terms of asserting the balance between the classification for the classes. Smaller values on the x-axis of the plot indicate lower false positives and higher true negatives. Larger values on the y-axis of the plot indicate higher true positives and lower false negatives. The correlation among the models

Table 4.4. Features of the UNSW NB15 dataset: flow-related (top), basic (next), content-related (next), time-related (next), and connection (bottom).

#	Name	Type	Description
1	srcip	N	Source IP address
2	sport	I	Source port number
3	dstip	N	Destination IP address
4	dsport	I	Destination port number
5	proto	N	Transaction protocol
6	state	N	The state and its dependent protocol, e.g. ACC, CLO, else (-)
7	dur	F	Record total duration
8	sbytes	I	Source to destination bytes
9	dbytes	I	Destination to source bytes
10	sttl	I	Source to destination time to live
11	dttl	I	Destination to source time to live
12	sloss	I	Source packets re-transmitted or dropped
13	dloss	I	Destination packets re-transmitted or dropped
14	service	N	http, ftp, ssh, dns .., else (-)
15	sload	F	Source bits per second
16	dload	F	Destination bits per second
17	spkts	I	Source to destination packet count
18	dpkts	I	Destination to source packet count
19	swin	I	Source TCP window advertisement
20	dwin	I	Destination TCP window advertisement
21	stcpb	I	Source TCP sequence number
22	dtcpb	I	Destination TCP sequence number
23	smeansz	I	Mean of the flow packet size transmitted by the src
24	dmeansz	I	Mean of the flow packet size transmitted by the dst
25	trans depth	I	the depth into the connection of http request/response transaction
26	res bdy len	I	The content size of the data transferred from the server's http service
27	sjit	F	Source jitter (mSec)
28	djit	F	Destination jitter (mSec)
29	stime	T	record start time
30	ltime	T	record last time
31	sintpkt	F	Source inter-packet arrival time (mSec)
32	dintpkt	F	Destination inter-packet arrival time (mSec)
33	tcprtt	F	The sum of 'synack' and 'ackdat' of the TCP.
34	synack	F	The time elapsed between the SYN and the SYN ACK packets of the TCP.
35	ackdat	F	The time between the SYN ACK and the ACK packets of the TCP.
36	is sm ips ports	B	If source (1) equals to destination (3)IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
37	ct state ttl	I	N. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct flw http mthd	I	N. of flows that has methods such as Get and Post in http service.
39	is ftp login	B	If the ftp session is accessed by user and password then 1 else 0.
40	ct ftp cmd	I	No of flows that has a command in ftp session.
41	ct srv src	I	N. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct srv dst	I	N. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct dst ltm	I	N. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct src ltm	I	N. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct src dport ltm	I	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct dst sport ltm	I	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct dst src ltm	I	No of connections of the same source (1) and the destination (3) address in 100 connections according to the last time (26).

metrics was calculated having as input data the specific figures for these metrics for each of the models produced. A model has been obtained for each point of the grid search space (explained next).

6 Approach and Parameters

Our approach can be summarised as follows:

- To define the values of the model's parameters for the two datasets, we used a full grid search approach, i.e., we did a search over the a hyperparameter space subset using H2O's grid search functionality for a full search. The range of discrete values used for the parameters in the search is in Table 6.1.
- The models are ranked via the validation logloss minimization criteria;
- The best model is set to be the model with the minimum validation logloss value;
- A correlation calculation of the models parameters versus logloss has been performed in order to seek which types of metrics can be used as proxy to estimate others;
- The two correlations were validated with a significance test.

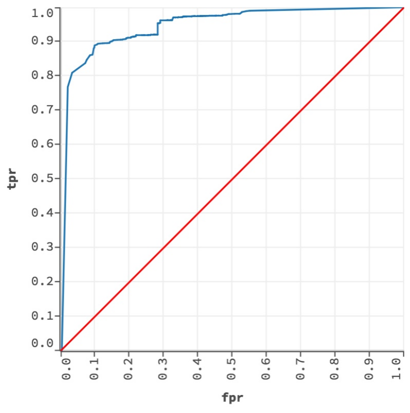
Table 6.1. Feature set used in the automatic training phase (discrete values) totalling $2^7 = 128$ different combinations.

XGBoost model parameter	**Values**
ntrees	5/10
max_depth	4/9
eta	0.1/0.5
sample_rate	0.5/0.9
gamma	0/0.001
reg_lambda	0/1.0
reg_alpha	0/1.0

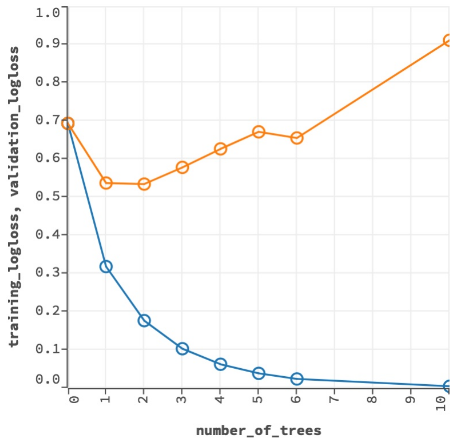
XGBoost, configured for acting as a tree booster, has been configured with the following parameters,[7] explicitly defined in the training phase with the ranges listed in Table 6.1:

- *ntrees*: number of trees to build;
- *max-depth* (default=6): maximum depth of a tree. Used to control over fitting as higher depth will allow model to learn relations very specific to a particular sample;
- *eta* (default=0.3): learning rate by which to shrink the feature weights. Shrinking feature weights after each boosting step makes the boosting process more conservative and prevents overfitting;
- *gamma* (default=0): a node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned;
- *subsample* (alias sample-rate) (default=1): row sampling ratio of the training instance;

[7] `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/xgboost.html#defining-an-xgboost-model`



(a) ROC curve (blue) for best model validation. AUC=0.9187.



(b) Logloss by number of trees (1-10); training (blue) and validation datasets (orange).

Fig. 7.1. UNB NSL KDD ROC curves and Logloss. It is relevant to note the divergence in b) regarding the validation Logloss that diverges denouncing a training dataset that is statistically dissimilar when compared with the testing dataset.

– *reg lambda* (default=1): L2 regularization term on weights (Ridge regression);
– *reg alpha* (default=0) L1 regularization term on weight (Lasso regression).

The values used in the grid search are listed in Table 6.1. All other parameters took the default values.

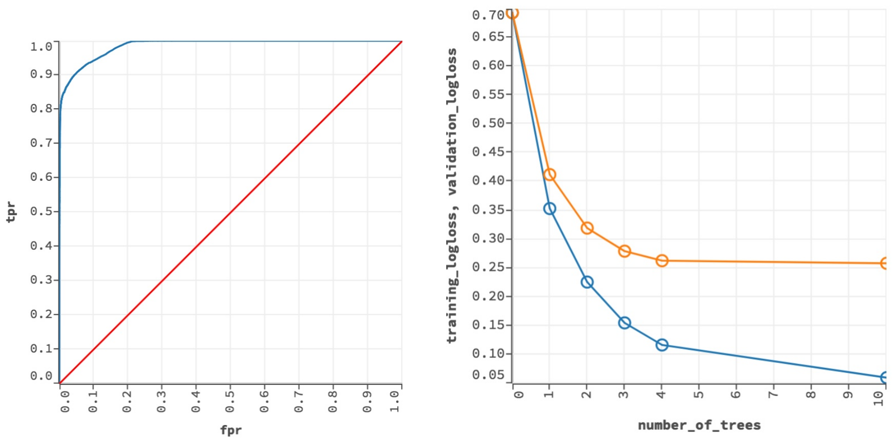
7 Evaluation

The objective of the experimental evaluation was to answer the three questions posed in the introduction. A number of performance metrics has been applied as per Table 7.1.

The parameters for the best models obtained for both datasets are depicted in Table 7.2. The main performance numbers are depicted in Figures 1(a), 1(a), 1(b), 2(a) and 2(b). Complementarily, Figures 4(a), 5(a), and 6(a) show performance values for the UNB NSL KDD dataset. Also Figures 4(b), 5(b), and 6(b) show the same for the UNSW NB15 dataset.

7.1 Logloss and AUC results

Regarding logloss, we can see in Figures 1(b) and 2(b) the fast convergence obtained with the training dataset as seen by number of trees in the x-axis (blue lines). Focusing on the validation dataset, for the KDD dataset the logloss figures did not converge to a minimum value (Fig. 1(b), orange line). The values obtained for AUC were high (see Table 7.1). As can be observed, the curve obtained is relatively well-balanced (Figure 1(a)). The predictor scored better with the UNSW NB15 dataset, in line with previous results by Moustafa [21].



(a) ROC curve (blue) for best model validation. Cross Validation AUC=0.9861.

(b) Cross Validation Logloss by number of trees (1-10); training (blue) and validation datasets (orange).

Fig. 7.2. UNSW NB15 ROC curves and Logloss. It is relevant to note a much lesser divergence in b) regarding the validation Logloss that diverges denouncing a training dataset that is statistically more resembling when compared with the testing dataset.

Table 7.1. Performance values for the best models obtained per dataset.

Dataset	KDD-NSL	UNSW NB15
Validation AUC	0.9187	0.9861
Validation Error	0.1136	0.0670
Validation Accuracy	0.8864	0.9334
Training Logloss	0.0577	0.0598
Validation Logloss	0.5816	0.2583

7.2 Interpreting the models

Figures 3(a) and 3(b) show that there are clusters of metrics highly correlated between each other (one central cluster with strong negative correlation and two triangle vertices clusters with strong positive correlation). The figures represent values of the Pearson correlation coefficient, probably the most widely used measure of the extent to which two variables are linearly related: X, Y: $\rho(X, Y) = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y}$. The metrics considered in this correlation test are [8]:*Validation R2, Validation Classification Error, Training Logloss, Cross Validation Logloss, Training RMSE, Cross Validation RMSE, Training Classification Error, Cross Validation Classi-*

[8] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/performance-and-prediction.html

Table 7.2. Relevant experimental best model features for the two datasets. It is relevant to remark the similarity between the two sets of values.

Parameter	NSL KDD	UNSW NB15
gamma	0.001	0
eta	0.5	0.5
sample_rate	0.9	0.9
max_depth	9	9
ntrees	10	10
reg_alpha	0	0
reg_lambda	0	0

fication Error, Validation AUC, Validation Lift, Validation RMSE, Validation Logloss, Training Lift, Cross Validation Lift, Training R2, Cross Validation R2 and *Training AUC*. The values of the Pearson correlation are shown with a p-test at a significance level of 0.001, so many correlations were not found significant and are shown crossed.

Two main clusters are evident for both the UNB NSL KDD (Fig. 3(a)) and UNSW NB15 cases (Fig. 3(b)), each with a high positive correlation and another with a null correlation among parameters. In Figures 7.7 and 7.8, the training features for both models show correlations with the training metric used, logloss. It is noticeable that for both training models the single most relevant features to affect the logloss results are the eta and $ntrees$ features. Some of the metric correlations also did not pass the significance test. These values are shown as crossed.

The divergent testing logloss in Figure 7.1 is caused by the fact that the testing dataset for KDD-NSL has considerable different statistical properties when compared with the train dataset. This situation can be identified by a learning curve for training loss that shows improvement and similarly a learning curve for validation loss that shows improvement, but a large gap remains between both curves.

7.3 Correlation validation

The p-test is used to check statistical significance of the correlation values obtained. Small p-values occur when the null hypothesis is false.

The validity of the test depends on the p-value used. Apparently small values of p like 0.01 still open a relatively large probability for the null hypotheses. Expanding on earlier work, especially Edwards et al. [12], it is shown that actual and relevant evidence against a null can differ by an order of magnitude from the p-value. The choice here is to choose a p-value equal or smaller than 0.001 as per Table 7.3.

7.4 How much information is hidden in the performance metrics?

The graphs in Figures 5(a), 4(a) and 6(a) can be understood as a way of empirically assessing the amount of information XGBoost manipulate in the datasets. This information amount can be assessed as an entropy value by means of the parallel with the entropy concept (equation 22): given the graphs are density functions, the entropy magnitude can be inferred as proportional to the graphs area, as per Equation 22.

$$H(X) = -\int_x p(x) \log p(x) \, dx \tag{22}$$

Table 7.3. Calibration table as a function of the p-values tested for significance.

$\mathbf{p}$	0.1	0.5	0.01	0.005	0.001
$\alpha(\mathbf{p})$	0.385	0.289	0.111	0.067	0.0184

It is straightforward to observe that the bigger this area is (entropy-like magnitude) for the training metrics the narrower and focused the values for the validation metrics are. The inverse is also observable. The relevance of this topic seems pertinent from the observations and deserves further study.

7.5 Discussion

Here we present a short explanation on how we answer the three questions posed in the introduction:

1. How effectively can XGBoost be used in the context of NIDSs? Our experiments show that XGBoost is a good approach for network intrusion detection, as the metrics we obtained can be considered very good, e.g., AUC circa 0.990 for the NB15 dataset case (Table 7.1).
2. How to optimize XGBoost automatic training for this application? We proposed a full grid search approach (Section 6). We used it to obtain the parameters for the detector for the two datasets (Table 7.2). The values for both datasets were similar, a fact that suggests that XGBoost is indeed a good approach as parameters can be reasonably stable for different datasets.
3. Among the parameters configured for XGBoost model training, can we find correlations with performance predictors, e.g., area under curve or logloss? The answer is positive, as shown in Figures 3(a) and 3(b).

8 Conclusions

We describe our approach to NIDS using XGBoost. The results show that the techniques proposed in the are effective fulfilling the objectives:

- We have shown and illustrated an effective method of automatic training for XGBoost models relying on only a limited fraction of the complete parameter space, attaining good performance metrics.
- The method allowed the most relevant model parameters for attaining such performance metrics to be identified, as illustrated in Figure 3(b) and 3(a).
- In Figures 7.7 and 7.8 the method proves its consistency by means of illustrating the consistent correlations among different performance metrics.

- The gain in model complexity is linear and it is remarkable when observing the most relevant features for the models performance as illustrated in Figure 7.9.

An evaluation with two well-known NIDS datasets has been performed using the XGBoost R implementation. Performances compatible with the best values obtained with other experiments has been confirmed. We proved the potential XGBoost has to be used in the context of network intrusion detection, in applying XGBoost to two representative datasets often used in machine learning research. A method for automatic best model parameter set identification is described. The method is based on a grid search approach, with the objective of logloss minimization as a criteria for model selection. By minimizing logloss and imposing closed grid search results have been obtained balancing computational time and model results.

Elements that contribute to the model interpretability were added, specifically elements linked to entropy of the resultant metrics (Section 7.4). Clear correlations were found among many important performance metrics. The correlation calculation executed with the performance metrics allowed to identify which metrics are correlated with others, and above all, identify which metrics do not correlate at all among each other.

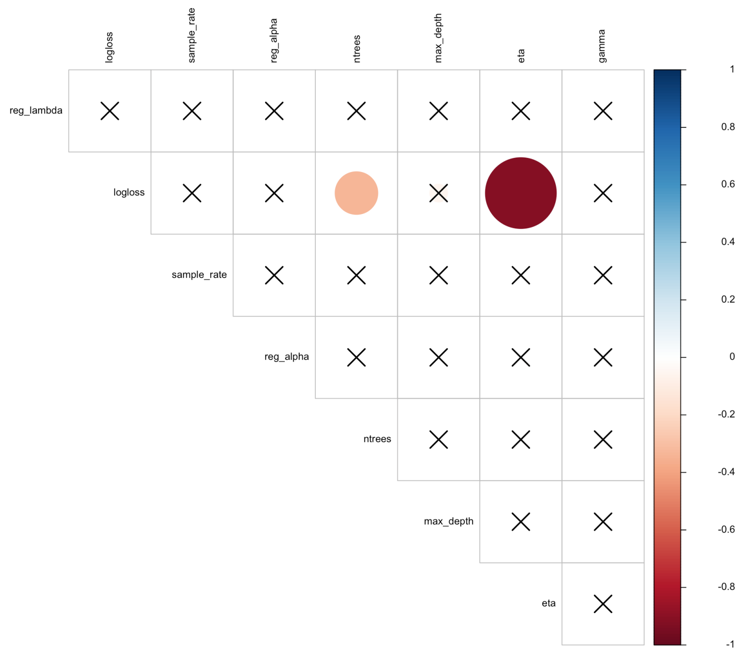
Acknowledgements. This research was supported by by the European Commission under grant agreement number 830892 (SPARTA) and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

References

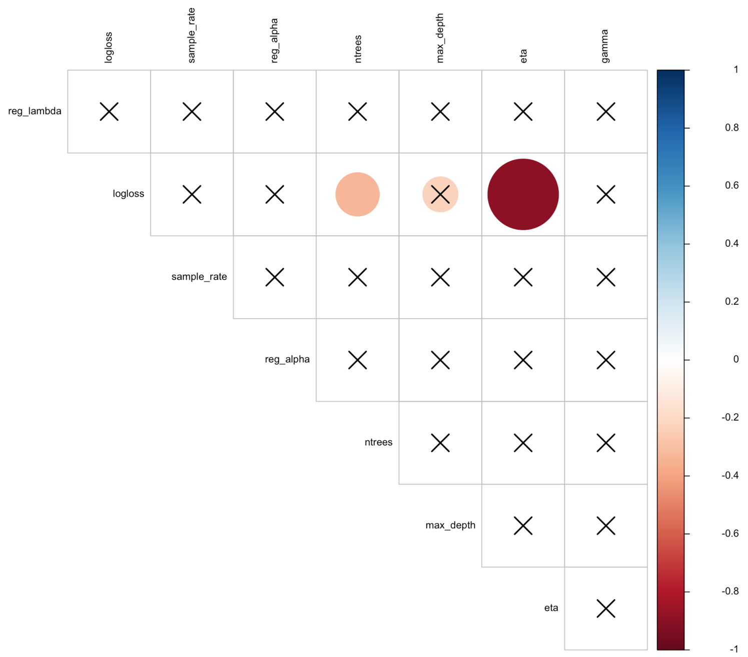
1. Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kégl, B., Rousseau, D.: How machine learning won the Higgs boson challenge. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (Apr 2016)
2. Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kégl, B., Rousseau, D.: The Higgs boson machine learning challenge. In: NIPS 2014 Workshop on High-energy Physics and Machine Learning. pp. 19–55 (2015)
3. Aiello, S., Kraljevic, T., Maj, P.: h2o: R Interface for H2O (2015), `https://CRAN.R-project.org/package=h2o`, r package version 3.6.0.8
4. Amaral, P., Dinis, J., Pinto, P., Bernardo, L., Tavares, J., Mamede, H.S.: Machine learning in software defined networks: Data collection and traffic classification. In: IEEE 24th International Conference on Network Protocols (ICNP). pp. 1–5 (Nov 2016)
5. Bansal, A., Kaur, S.: Extreme gradient boosting based tuning for classification in intrusion detection systems. In: Advances in Computing and Data Sciences. pp. 372–380. Springer (2018)
6. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30**(7), 1145–1159 (Jul 1997)

7. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794 (2016)
8. Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W., Peng, J.: XGBoost classifier for DDoS attack detection and analysis in sdn-based cloud. In: BigComp. pp. 251–256. IEEE Computer Society (2018)
9. Debar, H., Dacier, M., Wespi, A.: A revised taxonomy of intrusion detection systems. Annales des Télécommunications **55**(7), 361–378 (2000)
10. Dhaliwal, S., Nahid, A.A., Abbas, R.: Effective intrusion detection system using XGBoost. Information **9**(7), 1–24 (2018)
11. Dias, L.F., Correia, M.: Big data analytics for intrusion detection: an overview. In: Handbook of Research on Machine and Deep Learning Applications for Cyber Security, pp. 292–316. IGI Global (2020)
12. Edwards, W., Lindman, H., J. Savage, L.: Bayesian statistical inference in psychological research. Psychological Review **70**, 193–242 (05 1963)
13. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**(8), 861 – 874 (2006)
14. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security **28**(1-2), 18–28 (Feb 2009)
15. Hanley, J., Mcneil, B.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology **143**, 29–36 (05 1982)
16. Higgs, P.W.: Broken Symmetries and the Masses of Gauge Bosons. Physical Review Letters **13**, 508–509 (1964)
17. Masnadi-Shirazi, H., Vasconcelos, N.: On the design of loss functions for classification: Theory, robustness to outliers, and savageboost. In: Proceedings of the 21st International Conference on Neural Information Processing Systems. pp. 1049–1056 (2008)
18. Mitchell, R., Chen, I.R.: A survey of intrusion detection techniques for cyber-physical systems. ACM Computing Surveys **46**(4), 55:1–55:29 (Mar 2014)
19. Mitchell, R., Frank, E.: Accelerating the xgboost algorithm using gpu computing. PeerJ Computer Science **3**, e127 (2017)
20. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems. In: 2015 Military Communications and Information Systems Conference. pp. 1–6 (Nov 2015)
21. Moustafa, N.: Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic. Ph.D. thesis, University of New South Wales, Canberra, Australia (2017)
22. Moustafa, N., Slay, J.: The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective **25**(1-3), 18–31 (2016)
23. Nielsen, D.: Tree Boosting With XGBoost – Why Does XGBoost Win "Every" Machine Learning Competition? Master's thesis, NTNU (2016)
24. Sharafaldin., I., Lashkari., A.H., Ghorbani., A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. pp. 108–116. INSTICC (2018)
25. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security **31**(3), 357–374 (May 2012)
26. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications. pp. 53–58 (2009)

27. The CMS Collaboration: Evidence for the direct decay of the 125 GeV Higgs boson to fermions. Nature Physics **10** (Jun 2014)
28. The CMS Collaboration: Observation of the Higgs boson decay to a pair of τ leptons with the CMS detector. Physics Letters **B779**, 283–316 (2018)
29. Wheeler, P., Fulp, E.: A taxonomy of parallel techniques for intrusion detection. In: Proceedings of the 45th Annual Southeast Regional Conference. pp. 278–282 (2007)

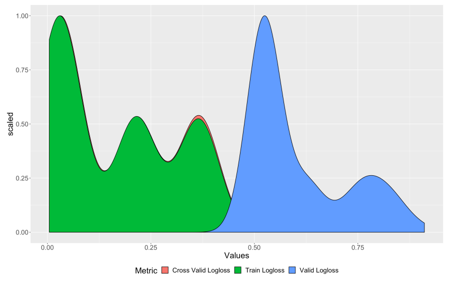


(a) UNB NSL KDD.

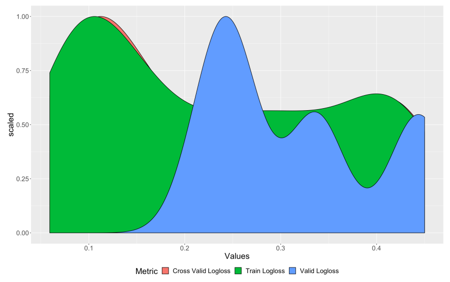


(b) UNSW NB15.

Fig. 7.3. Pearson correlation for the model variables. Correlation values found to be not significant are crossed. The magnitude of the correlation links to the size of the circles. Red colors are negative and blue positive. Values processed for all the iterated models with a significance level for the $p - test = 0.001$.

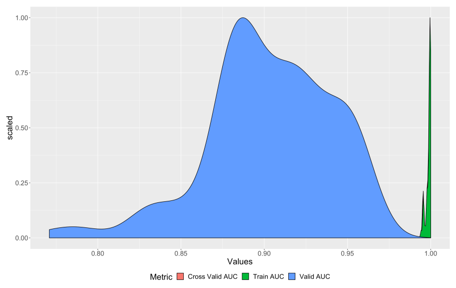


(a) UNB NSL KDD.

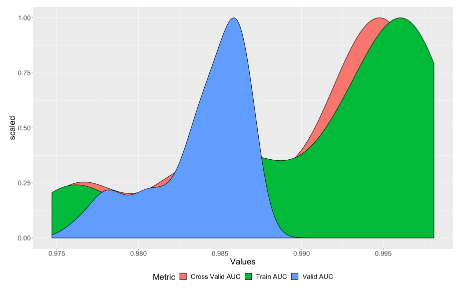


(b) UNSW NB15.

Fig. 7.4. Logloss values dispersion. Logloss has a relatively large spread. Cross Validation logloss mostly superimposes with Train logloss for the NB15 case denoting more similitude in terms of statistical characteristics between the train and test datasets.

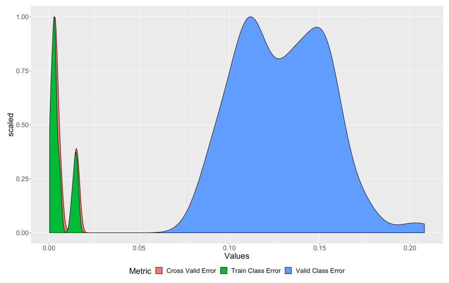


(a) UNB NSL KDD.

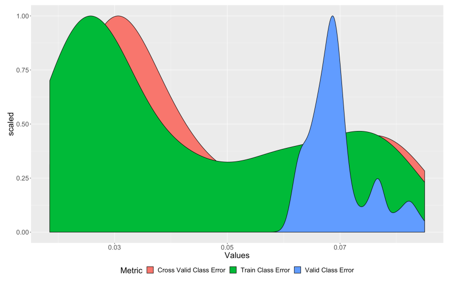


(b) UNSW NB15.

Fig. 7.5. AUC values dispersion. Validation AUC has a large spread for NSL KDD. On the opposite Cross Validation AUC and Training AUC have low spread. Again this different behaviour reveals differences between the test and train datasets for KDD, which are not so marked for the NB15 case.



(a) UNB NSL KDD.



(b) UNSW NB15.

Fig. 7.6. Classification error values dispersion. Note the large spread for Train Error and Cross Validation error in the NB15 case. On the contrary the Validation error has a much smaller dispersion. This suggests a different distribution subjacent to the dataset features.

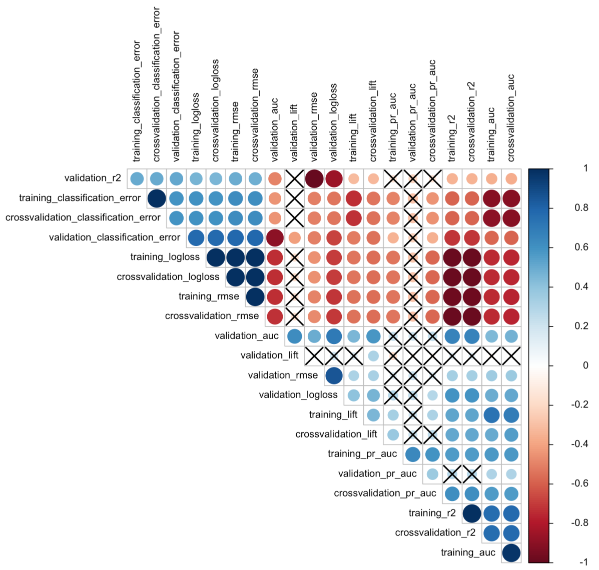


Fig. 7.7. UNB NSL KDD Pearson correlation tested against significance at 0.001 level. Non-significant correlation values are crossed.

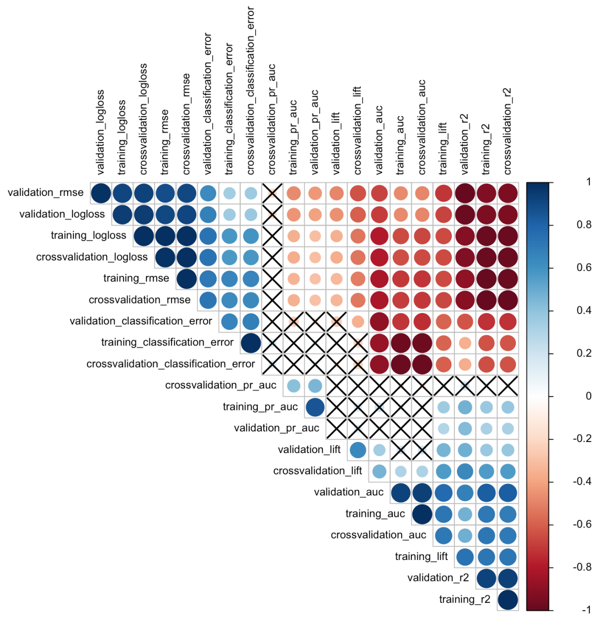
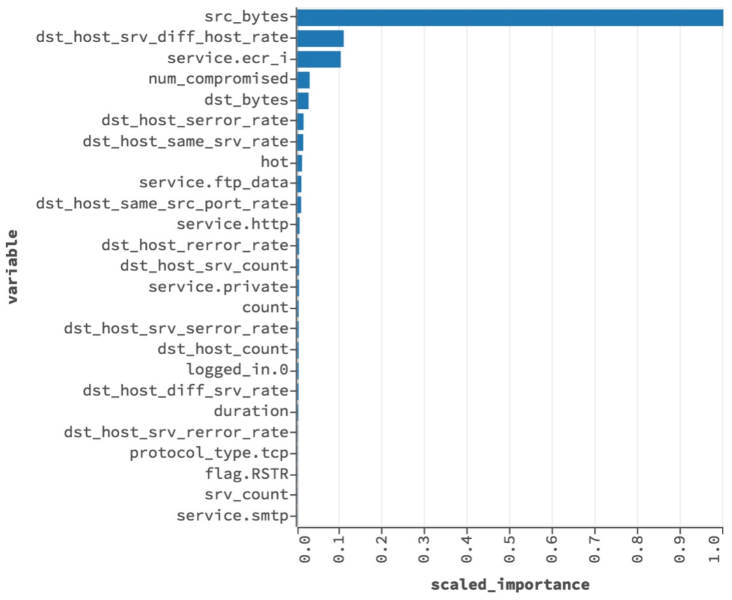
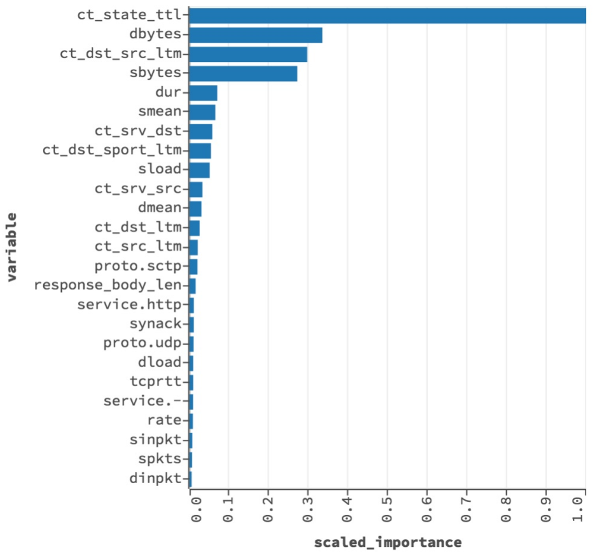


Fig. 7.8. NB15 Pearson correlation tested against significance at 0.001 level. Non-significant correlation values are crossed.



(a)



(b)

Fig. 7.9. Variable importance for the best model relative to NSL-KDD and NB15 datasets. The most commonly used threashold to discard irrelevant features is 0.1 in the scaled importance axis. For both cases the expected gains in complexity model if using a reduced set of features is remarkable.