Feature Set Tuning in Statistical Learning Network Intrusion Detection

Arnaldo Gouveia[1,2] Miguel Correia[2]
[1]Portugal Telecom [2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Lisboa, Portugal

***Abstract*—The detection of security-related events using machine learning approaches has been extensively investigated. In particular, machine learning applied to network intrusion detection systems (NIDS) has attracted a lot of attention due to its good generalization and unknown attack detection capabilities. A number of classification techniques have been used for this purpose, revealing good generalization properties. In this paper we go one step further by evaluating the performance of NIDSs when feature set tuning and reduction are realized. We evaluate a number of state of the art learning algorithms that are raising much interest but have not been used for intrusion detection yet. We compare a representative set of algorithms: Ada, ROC-based learners, two types of Classification Trees, Boosted Logistic Regression, Generalized Linear Models, Gradient Boosting Machines, and Neural Networks. The main objective is to reduce the number of features used – thus also the size of the data processed – to improve speed while maintaining adequate accuracy.**

I. Introduction

With the continuous growth in number and impact of network attacks, *network intrusion detection systems* (NIDS) are increasingly becoming critical security elements. From a research standpoint, although investigated for many years [16], NIDSs continue to get a lot of attention due to their practical interest regarding effective detection of malicious attacks, while processing large data volumes with machine learning algorithms [17], [30].

Anomaly-based network intrusion detection is a particularly promising approach as it allows detecting previously unknown attacks. This approach resorts to machine learning to create models of normal behavior, then detecting deviations. In such IDSs, attacks are detected when anomalies are identified. The generalization machine learning algorithms achieve in the learning phase allows the identification of anomalies even if the normal traffic observed in runtime is not identical to the traffic of the learning phase. False positives and false negatives are the cost of this generalization capability.

Deep learning is currently a hot topic in machine learning [1], [2]. Deep learning techniques have been achieving excellent results in recognition of speech, faces, and images in general, to name just a few examples [1], [22]. Although initially the term deep learning referred to neural networks with many layers ("deep"), today other algorithms apparently very different are known to have similar behavior and also put under the term. This paper also explores the use of other algorithms seldom employed in NIDSs: Generalized Linear Models (GLM) and Gradient Boosting Machines (GBM). Other more commonly referenced algorithms are also compared, namely Classification Trees and Neural Networks.

978-1-5090-3216-7/16/$31.00

The goal of this comparison is evaluating the potential of *feature set tuning* to improve detection speed while maintaining accuracy at acceptable levels. This is especially important in times in which NIDSs have to process the high traffic loads that travel our networks. Feature selection is an important part of the process of dataset tuning. Therefore it is important for machine learning-based NIDS development in order to save time while providing an insight into the underlying feature details relevant to the classification process. It is a well-known fact that not all traffic features (or attributes) are equally useful to detect attacks/intrusions [14], [27]. Therefore feature set tuning allows reducing the feature set, improving the training dataset with the goal of obtaining speed gains while maintaining an acceptable accuracy [27].

Our results show that after tuning all algorithms present similar metrics, although the number of features reduced depends on the algorithm. GBM has achieved the highest feature reduction, closely followed by RPART.

The main contributions of this paper are: (1) a comparison of intrusion detection performance with a set of relevant machine learning algorithms; (2) a study showing how feature set tuning allows reducing datasets and maintaining accuracy at similar acceptable levels. In this paper, we test tune the UNB ISCX Intrusion Detection Evaluation Dataset while validating a number of representative machine learning algorithms from the literature.

II. The Dataset

The UNB ISCX Intrusion Detection Evaluation Dataset was developed in an attempt to provide a quality dataset for network intrusion detection research [23]. The approach for defining this dataset involved identifying features that would allow effective detection, while minimizing processing costs.

Cost-based models have often been used regarding feature definition in fraud based detection. Stolfo et al. [25] have shown that cost-based assertions developed for fraud detection can be generalized and applied to network intrusion detection as a criteria for feature finding. With this approach in mind, the authors defined a set of features intrinsically related to specific classes of traffic anomalies like Probing, Remote to Local, Denial of Service, and User to Remote attacks. By maximizing cost in specific cost models, a number of features have been identified by the authors and used in the UNB ISCX Intrusion Detection Evaluation Dataset. In this context the features chosen were the best candidates for maximizing the types of cost characteristic to intrusions: (1) damage cost: the amount of damage caused by an attack if intrusion detection is not attained. (2) challenge cost: the cost to act upon a potential

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

TABLE I. ATTACKS IN THE UNB ISCX TRAIN / TEST DATASETS (ALL ATTACKS FROM THE FIRST EXIST ALSO IN THE SECOND)

intrusion when it is detected; and (3) operational cost: the resources needed identify the attacks.

The UNB ISCX dataset is composed of sequences of entries in the form of records labeled as either *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-layer 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 the UNB ISCX dataset fall into four classes:

Denial of Service Attacks (Dos). In this class of attacks the attacker renders computing or memory resources too busy or too full to handle legitimate requests or denies legitimate users access to a machine, e.g., land, syn flood, etc.

User to Root Attacks (U2R). This is a class of attacks in which the attacker starts out with access to a normal user account on the system and is able to exploit some vulnerability to gain unauthorized root access to the system. e.g., loadmodule or perl.

Remote to Login Attacks (R2L). This type of attack occurs when an external attacker exploits some vulnerability to gain local access as a user in that machine, e.g., ftp write, http tunnel, password guess, etc.

Probing Attacks. These are attempts to gather information about any systems for the purpose of circumventing its security controls, e.g., network scans, port sweep, nmap, satan, mscan, etc.

The UNB ISCX 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 I, to allow evaluating the ability of algorithms to generalize. The train dataset has around 2.2 GB of data, whereas the test dataset has 0.8 GB.

Each record of the dataset is characterized by features that fall into three categories: basic, content, and traffic. These features are represented in Table II.

Feature	Detail
duration	length of the flow in seconds
protocol-type	type of the 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

TABLE II. FEATURES USED TO CHARACTERIZE EACH FLOW IN THE DATASET: BASIC (TOP), CONTENT (MIDDLE), TRAFFIC (BOTTOM).

	Actual Normal	Actual Anomaly
Predicted Normal	T_P	F_N
Predicted Anomaly	F_P	T_N

TABLE III. CONFUSION TABLE MODEL FOR METRICS

III. PERFORMANCE METRICS

We use a set of metrics to compare the algorithms and the effect of feature tuning. These metrics are mostly obtained from the confusion matrix (see Table III). A positive is the detection of a malicious flow, and a negative a non-detection. The detection of lack of detection can be right (true) or wrong (false).

Accuracy. Accuracy measures how well a classification test identifies an event class. The accuracy is the sum of true results (both true positives and true negatives) divided by the total number of observations (sum of all true positives, true negatives, false positives and false negatives): $Acc = (T_P + T_N)/(T_P + T_N + F_P + F_N)$. Accuracy ranges from 0 to 1, being the 1 the most favourable for a strictly balanced dataset.

No information rate. The no-information rate metric is the proportion of the most common class.

Kappa. The kappa statistic is a measure of agreement between

observations, where an observation is a validation round. We use this metric as defined by Cohen [6]: $\kappa = (P_o - P_e)(1 - P_e)$, where P_o is the observed accuracy and P_e the expected accuracy under random agreement [29]. The most favourable case for $kappa$ is perfect agreement, which would equate to a kappa value of 1.

Sensitivity (or True Positive Rate). Sensitivity is the probability that a test will indicate a positive condition among the set of positives: $T_{PR} = T_P/(T_P + F_N)$. Sensitivity ranges from 0 to 1 with 1 being the most favorable case for a strictly balanced dataset.

Specificity (or True Negative Rate). Specificity measures the proportion of negatives that are correctly identified as such: $T_{NR} = T_N/(T_N + F_P)$. Specificity ranges from 0 to 1, with 1 being the most favorable case for a strictly balanced dataset.

Positive Predicted Value. The Positive Predicted Value (PRV) is the proportion of true positive results referenced to the sum of true positives and false positives results: $P_{PV} = T_P/(T_P + F_P)$. PRV ranges from 0 to 1, with 1 being the most favorable case for a strictly balanced dataset.

Negative Predicted Value. The Negative Predicted Value (NPV) is the proportion of true negative results referenced to the sum of true negatives and false negative results: $N_{PV} = T_N/(T_N + F_N)$. NPV ranges from 0 to 1, with 1 being 1 the most favorable case for a strictly balanced dataset.

Positive Likelihood Ratio. In medicine, likelihood ratios are used for confirming a diagnostic test. They use the sensitivity and specificity of the test to determine whether a test result confirms the probability that a condition (such as a disease state) exists [28]. In intrusion detection this metric can be used as a confirmation that positive indicators are supported by malicious and intentional activity and it has been used in IDS related research [12]. It is given by $LR_+ = Sensitivity/(1 - Specificity)$. Values greater than 1 and greater confirm the existence of intentional malicious behaviour activity with ever increasing probability as LR_+ grows.

Balanced Accuracy. Balanced accuracy is given by $cT_P/(T_P + F_P) + (1 - c)T_N/(T_N + F_N)$, where c belongs to the interval $[0, 1]$ translating the imbalance (c and $1 - c$ are in practice the priors). If the classifier performs equally well on either class, this term reduces to the conventional accuracy (number of correct predictions divided by number of predictions): $T_P/(2(T_P + F_P)) + T_N/(2(T_N + F_N))$. Balanced Accuracy ranges from 0 to 1, with 1 being the most favorable case for a strictly balanced dataset.

IV. The Method

The core of the experiment is a comparison of a set of representative machine learning algorithms in the context of network intrusion detection. The steps followed were:

- Selection of a dataset of network traffic designed for network intrusion detection evaluation;
- Tune the training of the algorithms and extract the subset of the most relevant features using the R `caret` package `varImp()` function. When no tuning parameters are provided by `caret`, default values are used;
- Train and test with a feature reduced dataset.

We start with the first and leave the rest for Section V.

A. Feature Importance Determination

From an optimization perspective it is relevant to know what is the importance of the various features (or variables) used in the learners in terms of their contribution to the final performance result. A number of approaches have been used in the past for this purpose, namely Correlation-based Feature Selection, Information Gain, Gain Ratio, and the ROC based variable importance [3]. In this paper we investigate the use of the area under curve (AUC) of the receiver operating characteristic (ROC) curve as a performance measure for NIDS based on some machine learning learners. The variable importance list has been obtained with the `varImp()` function of the `caret` R package, which uses a ROC AUC maximization approach. Feature $\#15$ of the original dataset was discarded because it was constant in both train and test datasets. The importance scale is presented in a percentual relative range for the 20 most relevant features in each algorithm.

B. Model Independent Feature Importance Selection

If there is no model-specific way to estimate feature importance (or the argument `useModel = FALSE` is used in `varImp()`), the importance of each feature is evaluated individually using a metric-based approach. This approach is model-independent. In face of this argument the `useModel = FALSE` option has been the choice for running the `varImp()` function in the models in which this option applies, namely Classification Trees and Random Forests.

ROC curve analysis is the approach used for reducing the feature set. For 2-class problems like intrusion detection (classes attack/no-attack), a series of cutoffs is applied to the algorithm data to predict the class. The sensitivity and specificity are computed for each cutoff and the ROC curve area is computed as the measure of variable importance. For a specific class, the maximum area under the curve across the relevant pair-wise AUCs is used as the variable importance metric. The model selected as the best is the candidate with the highest accuracy. If more than one tuning parameter is optimal then the function will try to choose the combination that corresponds to the least complex model [15].

C. Computational Environment

The data preparation phase, involving feature preprocessing, has been done using WEKA 3.6. WEKA stands for the Waikato Environment for Knowledge Analysis, which was developed at the University of Waikato of New Zealand. All the remaining experiments were performed with the R environment for statistical computing supported by the R Foundation for Statistical Computing [19]. R is widely used among statisticians and data miners for developing statistical software and data analysis. We used the following R packages: caret 6.0-68, rpart 4.1-10, neural net 1.32, C50 0.1.0-24, gbm 2.1.1, pROC 1.8 and rooc 1.2.

V. The Algorithms

The list of learners we considered is not exhaustive, but it contains a representative subset of good performance learners. This section presents the study itself. For each algorithm we explain briefly how it works and how it was tuned in order to obtain results as good as possible. The implementations of the algorithms used were those in R and its packages.

A. Ada (Boosted Classification Trees)

Ada is used in conjunction with other (weak) learning algorithms to improve their performance using boosting [10]. In this case we employ Ada as a booster of classification trees, used as weak learners.

In Boosted Classification Trees the training events that are misclassified (a signal event fell on a background leaf or vice-versa) have their weights increased (boosted), and a new tree is formed. This procedure is then repeated for the new tree. In this way many trees are built up. The score from the m_{th} individual tree T_m is taken as $+1$ if the event falls on a signal leaf and -1 if the event falls on a background leaf. The final score is taken as a weighted sum of the scores of the individual leaves [20].

Tuning. Cross-Validated (10 fold) re-sampling (also 10 fold) has been used. For classification using package `ada` the tuning parameters are: `Number of Trees` (iter), `Max Tree Depth` (maxdepth) and `nu` (learning rate). The Tuning parameter `nu` was held constant at a value of 0.1. Accuracy was used to select the optimal model using the largest value. The final values used for the model were `iter = 150, maxdepth = 3` and `nu = 0.1`.

B. Ranking Classifier

This is a classification method based on receiver operating characteristics (ROC). The method value rocc is chosen by `caret` from package `rocc` with the tuning parameter `xgenes`. Briefly speaking, features are selected according to their contribution to the ranked AUC value using the training set. The function performs classification by leave-one-out-cross-validation (LOOCV) using the ROC based classifier: features are combined to a group by the mean ROC value expression. Afterwards these samples are ranked according to their contribution to the AUC value. The feature group that yields optimal accuracy in the training samples is then used to classify new samples.

Tuning. The only tuning parameter available in `caret` is the number of variables retained in each LOOCV cycle: `xgenes`. Accuracy was used to select the optimal model using the largest value. The final value used for the model was `xgenes = 2`.

C. Classification Trees

Classification and regression trees were first described by by Brieman et al. [5]. Classification or Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. Decision trees create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The deeper the tree, the more complex the decision rules and the fitter the model [5]. We considered two algorithms: C5.0 and Rpart.

C5.0. This is a classification tree model that works by splitting the sample based on the field that provides the maximum information gain. The C5.0 model can split samples based on the largest information gain. The sample subset that is get from the former split is split after. The process will continue until the sample subset cannot be split. Finally a pruning process is executed examining the lowest level split upwards. Those sample subsets that do not have a significant contribution to the model information gain will be dropped.

Tuning. Cross-Validation (10 fold) has been used. The option for method definition in `caret` has been `method = C5.0Tree`. This method has no available tuning parameters in `caret`.

Rpart. Rpart can be generated through the `rpart` package. `Rpart` builds classification or regression models of a very general structure using a two stage procedure; the resulting models can be represented as binary trees. One of the decision trees implementation used was the one in R's `rpart` package.

Tuning. Cross-Validated (10 fold) re-sampling (3 fold) has been used. The option for `caret` has been `method = rpart2`. This method in `caret` has only available as tuning parameters `maxdepth` (maximum tree depth). Accuracy was used to select the optimal model using the largest value. The final value used for the model was `maxdepth = 3`.

D. Boosted Logistic Regression

Logistic regression was developed by Duncan and Walker [26] and Cox [7]. The Boosted Logistic regression learner has been described first hand by Friedman et al. [10]. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating the classes using a logistic function, which is the cumulative logistic distribution. Boosting sequentially applies a classification algorithm to weighted versions of the training data and then takes a weighted majority vote of the sequence of classifiers thus produced. In the context of `caret` package it can be used as a classifier.

Tuning. Cross-Validated (10 fold) re-sampling (also 10 fold) has been used. The method `LogitBoost` in `caret` admits `nIter` (number of boosting terations) as tuning parameter. The optimal number of iterations found for optimizing the accuracy figure was 31.

E. Generalized Linear Models

Generalized linear models (GLM) are a generalization of linear regression. Linear regression models the dependency of a response y on a vector of features $x(y \sim x^T\beta + \beta - 0)$. These models are built with the assumptions that y has a Gaussian distribution with a variance of σ^2 and the mean is a linear function of x with an offset of some constant $\beta - 0$, i.e., $y = \mathcal{N}(x^T\beta + \beta - 0; \sigma^2)$. These assumptions can be overly restrictive for real-world data that does not necessarily have a Gaussian distribution. GLM generalizes linear regression in the following way: it adds a non-linear link

function that transforms the expectation of response variable, so that $link(y) = x^T\beta + \beta - 0$ and allows variance to depend on the predicted value by specifying the conditional distribution of the response variable or the family argument.

Tuning. Cross-Validated (10 fold) re-sampling (also 10 fold) has been used. The GLM model, specified by "`method = glm`", offers no tuning parameters in `caret`.

F. Gradient Boosting Machines

Gradient Boosting Machines (GBM) is an algorithm designed to produce a model built by an ensemble of weak prediction models. It builds the model in a stage-wise fashion and is generalized by allowing an arbitrarily differentiable loss function. GBM fits consecutive trees as weak predictors where each solves for the net error of the prior. The idea of gradient boosting originated in Breiman's observation that boosting can be interpreted as an optimization algorithm using a suitable cost function [4].

Tuning. Cross-Validated (10 fold) re-sampling (also 10 fold) has been used. The tuning parameters offered by the `caret` package are: `n.trees` (number of boosting iterations), `interaction.depth` (maximum tree depth), `shrinkage` (shrinkage), `n.minobsinnode` (minimum terminal node size). The maximum depth of a tree is a tree specific parameter used to control over-fitting as higher depth will allow the model to learn algorithmic rules very specific to a particular sample. In the tuning phase a set of maximum depth of 2, 3 and 4 have been used.

A non-negative integer that defines the number of trees, (`trees`) has been tested with two values: 5 and 10. The learning rate, as a boosting parameter, has been set to `0.1`. This determines the impact of each tree on the final outcome. Lower values are generally preferred as they make the model robust to the specific characteristics of the tree allowing it to generalize well, although with a cost in terms of processing time. Several good and approximate models have been obtained. We presented the results for one with number of `trees=10, mean depth=4, min leaves=15` and `max leaves=16`.

G. Neural Networks

The variety of deep learning architectures is vast and includes examples such as Autoencoders, Multilayer Perceptron, Recurrent Neural Networks (RNNs), Restricted Boltzmann Machines (RBMs), Self Organizing Maps (SOMs) and Convolutional Neural Networks. In the scope of this category of classifiers a more shallow option of an one hidden layer neural net has been used, namely the `nnet` model of the `caret` package.

Tuning. Ten fold cross-validation was used. Tuning parameter `size` was tuned with values of 8 and 16. The value for `decay` was tuned with values of `1e-04` and `1e-03`. Accuracy values were used to select the optimal model using their largest value after 500 iterations. The final values chosen for the model were `size = 16` and `decay = 1e-04`.

VI. Result Analysis and Discussion

The results for all algorithms are presented in Table IV for the case of no feature selection, and Table V for the case of a reduced set of features. The values for accuracy, balanced accuracy, kappa and positive likelihood ratio are shown in Figures 1, 2, 3 and 4. As the dataset has class priors near 50%, the balanced accuracy exhibits the same behaviour as the accuracy. It is commonly accepted that a good model has to have low training error and low generalization error (or test error). However the need for generalization often leads to accuracy levels that are not near 1 for most algorithms, for a balanced dataset.

Overall speaking the accuracy and balanced accuracy exhibit reasonably good values, although in bands that potentially preclude the onus of overfitting. On the other hand, the values of kappa fall in ranges that are considered moderate (range 0.41 to 0.60) to substantial agreement (range 0.61 to 0.80) [29].

Domingos [8] states a number of conditions for learners to produce useful results. Among others we have the following: (1) Overfitting is to be avoided. For instance the presence of noise may aggravate overfitting, i.e., existence of mislabeled instances contradicting the class labels of another similar record may be a liability in this sense. (2) Representativeness does not imply learnability. Lack of representative instances in the training data may be useless but at the same time there may be representative data that may not be learnable. Therefore, features have to aid learnability. (3) Feature engineering is a key factor for obtaining good results.

As in our case the learner models do not overfit and the number of features is reduced, we may conclude that either the discarded features do not contribute to representativeness or they do not impact significantly the learner's performance because they do not contribute enough to learnability. As the features were engineered with specific cost-related criteria the latter seems to be a better candidate to explain the results. Also one may raise the question on how these set of classifiers may behave with other datasets. From our perspective the answer is already given by Domingos [8] when he states the conditions for a classifier to be useful.

There is an obvious similitude in the set of features and their relative importance concerning the Ada and ROC learners as seen in Table VI. This behaviour is most likely due to the known equivalence between Ada boosted learners and rank based learners (which use the same ROC AUC criteria) [21]. Although similarity verifies as expressed in Tables IV and V, performance results are not identical which is a direct consequence of the dissimilitude of the learners themselves.

It also noteworthy that there is virtually no difference between the results regarding Tables IV and V for the Boosted Logistic Regression (used as a classifier as remarked before), certainly due to specific properties of boosting. In this context Duchi et al. [9] have shown that for some types of Logistic Boosting the selective elimination of features had only a marginal effect on the test error.

Table VII shows in detail the performance differential from the complete feature set case to the reduced feature set case. For decision trees algorithms `RPART` and `C5.0` there

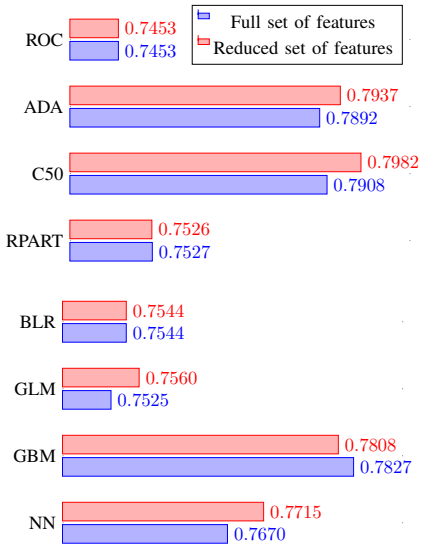

Fig. 1. Accuracy values.

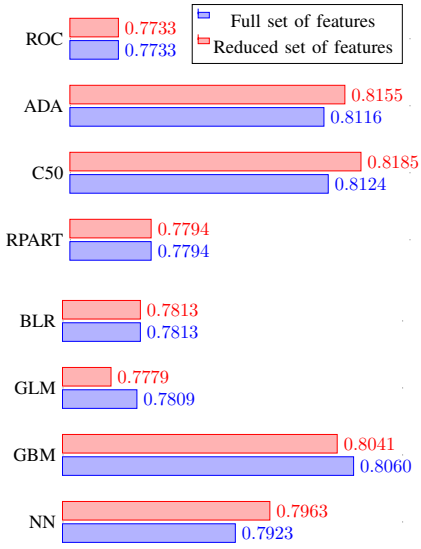

Fig. 2. Balanced accuracy values.

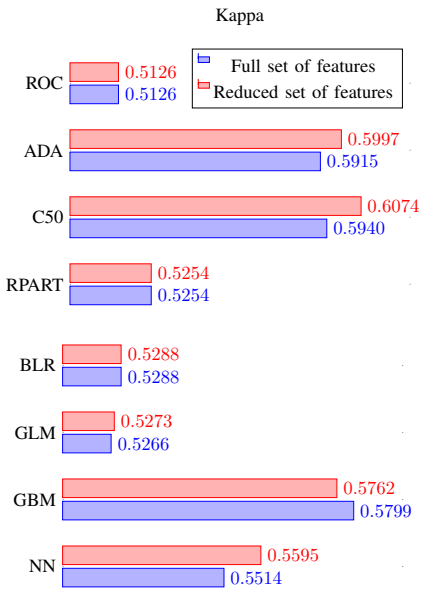

Fig. 3. Kappa values.

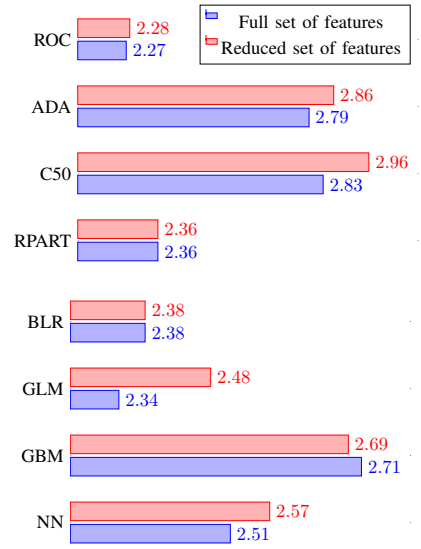

Fig. 4. Positive Likelihood Ratio values.

is clearly a lack of significantive change from the full set case to the reduced feature set case, with the exception of `Neg Pred Value` for `RPART`. This is likely due to the efficiency characteristics of decision trees, already noticed for small datasets and for the `C4.5` classifier (predecessor of the `C5.0` used here) [13].

In the case of neural networks (NN) feature reduction did not affect significantly the metrics obtained. It is known that neural networks are sensitive to the variety of train features used [11], [18], [24]. This relates to the setting of the internal network weights that are directly influenced by the input information. In our case as the number of features was reduced, the information for training is also less. As such this is not an expected result, although admittedly possible if the discarded features have brought no information.

The values for the Positive Likelihood Ratio, as illustrated in Figure 4 are of the same order of magnitude in all cases. In fact from the full feature set to the reduced feature set its relevance in confirming the assertion of the conclusion of positiveness supported on the other metrics is fundamentally maintained.

The effect of feature tuning was a reduction of the number of features. The number of features reduced for each algorithm is shown in the last row of Table VI. It is clear that similar results for similar performance metrics can be obtained at savings of a variable number of features (see also Tables IV and V). `GBM` has come up as the most feature-efficient learner, closely followed by `RPART`.

One common criteria for model acceptance is that the overall accuracy shall be higher than the no-information rate.

This latter metric was equal to 0.5709 in all algorithms (so it was omitted from the tables). It is remarkable that in any tested case the *p-value* or, equivalently, the probability that the accuracy results are obtained by chance are nearly null. This means that there is a significative relation between the features and their attributed class. In all cases the overall accuracy validates at over a 90% significance level with a *p-value* nearly null: 2.2^{-16}.

VII. Conclusions

Several conclusions can be extracted from the comparative analysis of the algorithms. A number of metrics have been used for this comparison, as seen in Tables IV and V. The set of algorithms tested proved to convey equivalent learning models and results under the same test conditions. This supports the conclusion of Shiravi at al. [23] regarding the quality of the UNB ISCX dataset. One of the main objections raised to this dataset was related to the disparity of results obtained. However, our results with the dataset seem to indicate similar results given the same experimental conditions with an extended variety of machine learning algorithms.

We have confirmed the validity of the ROC criteria for feature reduction. In this regard comparing the performance of classifiers with and without feature reduction proved favourable for the feature reduction case where performance metrics have shown to remain stable without loosing generalization power due to an eventual marked increase in accuracy. Examining the results from the feature reduction we observed that the resulting set of features and their importance varies considerably from learner to learner.

Our results show that feature optimality is classifier-dependent, at least for the ROC selection criteria. GBM has come up as the most feature-efficient learner, closely followed by RPART. Despite the disparity of results regarding features to discard, the learners have show substantial similitude regarding the metrics observed. This fact may hint at the value of ROC based ranking learners in choosing the best features case to case, learner to learner.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).

References

[1] G. Anthes. Deep learning comes of age. *Communications of the ACM*, 56(6):13–15, 2013.

[2] I. Arel, D. C. Rose, and T. P. Karnowski. Deep machine learning: A new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 5(4):13–18, 2010.

[3] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[4] L. Breiman. Arcing the Edge. Technical report, Statistics Department, University of California, Berkeley, June 1997.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[6] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37 – 46, 1960.

[7] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242, 1958.

[8] P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Oct. 2012.

[9] J. Duchi and Y. Singer. Boosting with structural sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 297–304. ACM, 2009.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[11] G. D. Garson. Interpreting neural-network connection weights. *AI Expert*, 6(4):46–51, Apr. 1991.

[12] J. Grana, D. Wolpert, J. Neil, D. Xie, T. Bhattacharya, and R. Bent. A likelihood ratio anomaly detector for identifying within-perimeter computer network attacks. *J. Netw. Comput. Appl.*, 66(C):166–179, May 2016.

[13] L. B. Holder. Intermediate decision trees. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, pages 1056–1062, 1995.

[14] N. Kayacik and M. Heywood. Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. In *The 3rd Annual Conference on Privacy, Security and Trust*, 2005.

[15] M. Kuhn. Building predictive models in R using the caret package. *R Project*, 2008.

[16] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, 1994.

[17] H. A. Nguyen and D. Choi. Application of data mining to network intrusion detection: Classifier selection model. In *11th Asia-Pacific Network Operations and Management Symposium*, pages 399–408, 2008.

[18] J. D. Olden and D. A. Jackson. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, 154(1):135–150, 2002.

[19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

[20] B. P. Roe, H.-J. Yang, and J. Zhu. Boosted decision trees, a powerful event classifier. In *Statistical problems in particle physics, astrophysics and cosmology. Proceedings, Conference*, pages 139–142, 2005.

[21] C. Rudin and R. E. Schapire. Margin-based ranking and an equivalence between adaboost and rankboost. *Journal of Machine Learning Research*, 10:2193–2232, Dec. 2009.

[22] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

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

[24] M. Stevenson, R. Winter, and B. Widrow. Sensitivity of feedforward neural networks to weight errors. *IEEE Transactions on Neural Networks*, 1(1):71–80, Mar 1990.

[25] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan. Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, pages 130–144, 2000.

[26] D. B. D. Strother H. Walker. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1/2):167–179, 1967.

[27] S. Suthaharan and T. Panchagnula. Relevance feature selection with data cleaning for intrusion detection system. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–6, March 2012.

[28] J. A. Swets. The relative operating characteristic in psychology. *Science*, 182(4116):990–1000, 1973.

[29] A. Viera and J. Garrett. Understanding interobserver agreement: The kappa statistic. *Family Medicine*, 37, 5 2005.

[30] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th ACM Annual Computer Security Applications Conference*, 2013.

Regression Metrics	ADA	ROC	C5.0	RPART	BLR	GLM	GBM	NN
Accuracy	0.7892	0.7453	0.7908	0.7526	0.7544	0.7525	0.7827	0.7670
Kappa	0.5915	0.5126	0.5940	0.5254	0.5288	0.5266	0.5799	0.5514
Specificity	0.6533	0.5756	0.6602	0.5906	0.5922	0.5806	0.6415	0.6135
Sensitivity	0.9699	0.9711	0.9646	0.9682	0.9703	0.9812	0.9706	0.9712
Neg Pred Value	0.9665	0.9636	0.9612	0.9611	0.9637	0.9763	0.9667	0.9659
Pos Pred Valuee	0.6777	0.6323	0.6809	0.6399	0.6414	0.6375	0.6705	0.6538
Likelihood Ratio(+)	2.79	2.27	2.83	2.36	2.38	2.34	2.71	2.51
Balanced Accuracy	0.8116	0.7733	0.8124	0.7794	0.7813	0.7809	0.8060	0.7923

TABLE IV. PERFORMANCE WITHOUT REDUCTION BY FEATURE SELECTION.

Regression Metrics vs. learner	ADA	ROC	C5.0	RPART	BLR	GLM	GBM	NN
Accuracy	0.7937	0.7453	0.7982	0.7527	0.7544	0.7560	0.7808	0.7715
Kappa	0.5997	0.5126	0.6074	0.5254	0.5288	0.5273	0.5762	0.5595
Specificity	0.6614	0.5756	0.6753	0.5906	0.5922	0.6234	0.6398	0.6216
Sensitivity	0.9696	0.9711	0.9617	0.9682	0.9703	0.9324	0.9684	0.9709
Neg Pred Value	0.9666	0.9636	0.9592	0.9612	0.9637	0.9246	0.9642	0.9660
Pos Pred Valuee	0.6828	0.6323	0.6900	0.6400	0.6414	0.6504	0.6689	0.6585
Likelihood Ratio(+)	2.86	2.28	2.96	2.36	2.38	2.48	2.69	2.57
Balanced Accuracy	0.8155	0.7733	0.8185	0.7794	0.7813	0.7779	0.8041	0.7963

TABLE V. SUMMARY PERFORMANCE WITH A REDUCED FEATURE SET DETERMINED BY FEATURE SELECTION.

Feature vs. Learner	ADA	ROC	C5.0	RPART	GLM	GBM	BLR	NN
dst-host-count	60.05	60.05	0	1.89	30.62	0	60.05	20.73
dst-bytes	100.00	100.00	46.67	0	0	26.50	100.00	0
count	83.64	83.64	41.89	15.61	67.87	7.56	83.64	55.17
src-bytes	99.92	99.92	100.00	100.00	0	100.00	99.92	63.93
dst-bytes	0	0	0	92.04	0	0	0	54.91
duration	0	0	1.35	0	15.57	0.82	0	20.10
srv-count	0	0	0	8.68	0	1.48	0	34.79
logged-in	87.21	87.21	45.44	0	0	0	87.21	24.64
num-shells	0	0	43.88	0	0	0	0	0
num-compromised	0	0	53.64	0	42.17	0	0	0
dst-host-srv-rerror-rate	0	0	0	0	0	0	0	0
dst-host-same-srv-rate	93.00	93.00	45.55	0	0	0.62	93.00	16.34
srv-serror-rate	77.25	77.25	0	0	34.93	0	77.25	11.17
wrong-fragment	0	0	0	0	100.00	0	0	13.70
hot	0	0	48.22	0	69.71	2.80	0	0
service	44.62	44.62	7.14	0	0	0	44.62	100.00
rerror-rate	0	0	0	0	0	0	0	13.59
serror-rate	80.78	80.78	0	0	0	0	80.78	11.57
urgent	0	0	0	0	0	0	0	0
dst-host-same-src-port-rate	44.34	44.34	55.57	0	63.76	4.11	44.34	34.36
is-guest-login	0	0	0	0	77.82	0	0	0
dst-host-srv-count	98.55	98.55	49.73	5.91	72.05	5.96	98.55	36.81
protocol-type	40.28	40.28	52.98	6.70	64.45	5.32	40.28	29.51
num-root	0	0	0	0	42.43	0	0	0
srv-rerror-rate	0	0	0	0	38.99	0	0	0
dst-host-rerror-rate	0	0	47.93	0	28.79	0	0	0
srv-diff-host-rate	45.80	45.80	0	0	27.78	0	45.80	0
num-file-creations	0	0	0	0	21.26	0	0	0
dst-host-srv-serror-rate	76.06	76.06	2.52	0	17.25	0	76.06	0
dst-host-serror-rate	81.80	81.80	0	0	16.59	0	81.80	11.07
same-srv-rate	94.68	94.68	43.92	81.85	15.27	0	94.68	0
diff-srv-rate	88.63	88.63	0	80.33	0	0	88.63	0
dst-host-srv-diff-host-rate	54.95	54.95	49.74	4.95	0	0	54.95	15.90
dst-host-diff-srv-rate	84.92	84.92	5.33	0	0	0	84.92	17.33
num-failed-logins	0	0	49.36	0	0	0	0	0
flag	91.11	91.11	47.75	75.56	47.30	0	91.11	88.48
Number of discarded features	20	20	20	29	20	30	20	20

TABLE VI. RELATIVE IMPORTANCE FOR EACH FEATURE BY LEARNER REPRESENTED IN A RELATIVE PERCENTUAL SCALE.

Regression Metrics vs. learner	ADA	ROC	C5.0	RPART	BLR	GLM	GBM	NN
Accuracy	1	0	1	0	0	0	0	0
Kappa	1	0	2	0	0	0	-1	0
Specificity	1	0	2	0	0	7	0	1
Sensitivity	0	0	0	0	0	-5	0	0
Neg Pred Value	0	0	0	0	0	-6	0	0
Pos Pred Value	1	0	1	0	0	2	0	0
Balanced Accuracy	0	0	1	0	0	0	0	0

TABLE VII. SUMMARY OF DIFFERENCES BETWEEN ORIGINAL AND FEATURE REDUCED DATASETS. VALUES ARE EXPRESSED IN PERCENTAGE. VALUES LESS THAN ONE ARE REPRESENTED AS NULLS.