MLNET - Machine Learning Models for Network Analytics

Pedro Casas
AIT - Austrian Institute of Technology
Vienna, Austria
pedro.casas@ait.ac.at

ABSTRACT

The application of Machine Learning (ML) models to the analysis of network measurement problems has largely increased in the last decade; however, there is still no clear best-practice or silver bullet approach to address these problems in a general context, and only adhoc and very tailored approaches have been evaluated so far. While deep-learning models have provided a major breakthrough in highly-dimensional problems such as image processing, it is difficult to say today which is the best model or most fitted category of models to address the analysis of large volumes of highly-dimensional data collected in operational networks. In this paper we evaluate and benchmark different ML models applied to the analysis of three different and assorted network measurement problems, including detection of network attacks, detection of smartphone-apps anomalies and QoE prediction in cellular networks. We consider an extensive battery of ML models, including both supervised and semi-supervised techniques, as well as ML ensembles such as bagging, boosting and stacking. Proposed models are evaluated using real network measurements coming from operational networks. Results suggest that both neural networks and decision-tree-based models provide in general better results in terms of accuracy and prediction, with a much smaller computation overhead for decision trees as compared to models based on neural networks or support vector machines. In addition, collaborative models, and in particular stacking models, are more robust and perform better than single ML models.

1 INTRODUCTION

Data-driven networking, i.e., the design and management of network systems by the analysis of network measurements, represents a key component for future network management. The high-volume and high-dimensionality of network data provided by current network measurement systems opens the door to the massive application of machine learning approaches to improve data-driven networking problems.

There is however a major challenge in applying machine learning models at large-scale for handling network measurements; selecting the best machine learning model for a specific problem is a complex task - it is commonly accepted that there is no silver bullet for addressing different problems simultaneously. Indeed, even if multiple models could be very well suited to a particular problem, it may be very difficult to find one which performs optimally for different data distributions and statistical mixes.

Deep-learning models are today widely used in multiple signal processing problems, particularly in image processing, where they have shown an outstanding performance. However, neural-networks based models, and particularly deep-learning models, have an inherent problem linked to model visibility and interpretation: a deep-learning model is a black-box which can automatically perform feature selection from input raw data and provide highly accurate predictions, but it is very difficult to understand their functioning. Indeed, it becomes very challenging to understand the reasons of a particular classification result, and in particular to understand the input features leading to such a result, as input features derived directly from a deep neural network architecture can be in general meaningless to a domain expert. This is one of the reasons why their application to networking problems is so far quite limited. In addition, deep-learning models are highly data-eager and training them is extremely costly in terms of computational power, which might term them unsuitable for different networking problems which require periodical re-training or where labeled data are difficult to get.

In this paper we pose ourselves a simple question: which type of machine learning model should be generally used in the analysis of network measurements? Intuition suggests that rule-based models could be in principle a good match for network analytics, as network protocols are highly structured and operate in a rule basis. We therefore present a comparative analysis of different machine learning models, applied to three specific network analytics problems: the detection of network attacks, the detection of network anomalies and the prediction of network performance in terms of end-use Quality of Experience (QoE).

We consider standard and well known machine learning models, which shall ease the interpretation of results and make them more applicable to common networking practitioners. These include decision-trees - single trees and random forests, naïve bayes models, neural networks, support vector machines and nearest neighbors models. We additionally consider collaborative or ensemble learning models, covering the three basic approaches to ensemble-learning: bagging, boosting and stacking. Rather than finding the best model to explain the data, ensemble learning methods build a set of models and then decide between them with some combinatorial approach, seeking model complementarity. Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In principle, if no single model covers the true prediction behind the data, an ensemble can give a better approximation of that oracle, true prediction model. An ensemble of models also exhibits higher robustness with respect to uncertainties in training data, which is highly beneficial for generalization of results. We believe that this study would enable a broader application of machine learning to data-driven networking problems, opening the doors to better and more cost-effective network analytics.

This paper builds on top of our recent early work on ensemble-learning models for network analytics [21], where we explore the application of ensemble-learning techniques to network security and anomaly detection, and on machine learning for QoE prediction [20]. The reminder of the paper is organized as follows. Sec. 2 presents a brief overview on the related work. In

Sec. 3 we briefly describe the evaluated machine learning models. Sec. 4 describes the evaluated network measurement and analytics problems and corresponding datasets. Sec. 5 presents the experimental results of the study, benchmarking the accuracy of the proposed models in the analysis of these problems. Finally, Sec. 6 concludes the paper.

2 STATE OF THE ART

The application of machine learning models to network measurement problems is largely extended in the literature. Traffic prediction and classification are two of the earliest machine learning applications in the networking field. In [2], authors provide a survey on different networking problems which have been addressed by machine learning approaches in the past, including objectives such as traffic prediction, traffic classification, network management, self-configuration, as well as performance analysis and prediction.

There are a couple of extensive surveys and papers on network measurement problems such as network anomaly detection [13, 14] - including machine learning-based approaches [11], machine learning for network traffic classification [18] and network security [15], as well as machine learning models for QoE modeling [19] and prediction [20].

The specific application of ensemble learning approaches is by far more limited. Even if it is generally observed in the practice that ensembles tend to yield better results than single models, only few papers have applied them to problems such as anomaly detection [16] and network security [17]. There is a recent surge on the application of deep learning models for network measurement problems, for example for network anomaly detection [10].

3 MACHINE LEARNING MODELS

In the context of supervised learning there are several approaches for predictive model training based on labeled data. The performance of a particular algorithm or predictor depends on how well it can assimilate the existing information to approximate the oracle predictor, i.e. the ideal optimal predictor defined by the true data distribution. However, knowing a priori which algorithm will be the best suited for a given problem is almost impossible in practice. One could say that each algorithm learns a different set of aspects of reality from the training datasets, and then their respective prediction capability also differs between problems.

In this paper we consider six standard machine learning models previously used in the literature for the analysis of network measurements, including: (i) decision-trees (CART), (ii) Naïve Bayes (NB), (iii) Multi-Layer Perceptron (MLP) Neural Networks, (iv) Support Vector Machines (SVM), (v) Random Forest (RF) and (vi) Nearest Neighbors (k-NN). We additionally compare three different approaches to ensemble-learning, including bagging, boosting (AdaBoost) and stacking (Stacking MV and GML). We briefly describe all these approaches next.

3.1 Decision Trees

Classification And Regression Trees (CART) [18] define a classification technique based on a tree graph, where inner nodes correspond to a condition on a feature and leaves are the outcome (i.e., the class). A CART represents a very popular classification algorithm due to its simplicity (it can be easily converted into a rule-based classification system) and readability (it can be graphically represented). The training follows a top-down greedy algorithm that works by iteratively splitting the nodes, using normally an information gain based metric as optimization criterion.

3.2 Naïve Bayes

Naïve Bayes (NB) is a very simple classifier based on Bayesian statistics [18]. Despite its simplicity, it is widely used as it is very efficient in a number of scenarios, especially in high-dimensional datasets. It works by assuming that features are mutually independent, which is not true in most cases, hence the adjective naïve. This assumption allows for an easy calculation of the class-conditional probabilities, using maximum likelihood estimation.

3.3 Neural Networks

Multi-Layer Perceptron (MLP) is an artificial neural network composed of multiple layers of neurons, each of them generally represented by a non-linear function [18]. The layers are fully connected in a feed-forward scheme. Each neuron employs an activation function that maps the weighted inputs to the output that is passed to the following layer. The weights, originally set to random values, are iteratively adjusted during the training phase, using back-propagation.

3.4 Support Vector Machines

Support Vector Machines (SVM) are non-probabilistic binary classifiers [18]. SVM is considered one of the most powerful supervised classification algorithm. It works by representing each feature vector in a multidimensional space and trying to find a linear separation (i.e., an hyperplane) for the classes. In some cases, however, a linear separation of the space is not possible, hence it uses the so-called kernel trick, which implicitly increases the dimensionality of the space, resulting in an easier separation in a much higher dimensional space, due to the increased sparsity.

3.5 Random Forrest

Random Forrest (RF) is an ensemble technique based on multiple instances of decision trees, each one based on a different part of the training set, randomly selected. These instances are called bootstrapped samples. The final outcome is generally decided by majority voting among all the bootstrapped samples.

3.6 k Nearest Neighbors

The k-Nearest Neighbors algorithm (k-NN) is a non-parametric approach used for either classification or regression. In both cases, the input consists of the k closest training examples in the feature space. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

3.7 Bagging and Boosting Algorithms

Bagging [6] decreases the variance of the prediction model by generating additional training data from the original dataset. Bagging trains each model in the ensemble using a randomly drawn subset of the training set, and each model in the ensemble is then combined in an equal-weight majority voting scheme. Increasing the training data size using a single input dataset does not improve the prediction accuracy, but narrows the prediction variance by strongly tuning the outcome.

Boosting [7] involves incrementally building an ensemble by training each new model instance based on the performance of the previous model. Boosting is a two-steps approach, where one first uses subsets of the original data to produce multiple models, and then *boosts* their performance by combining them, also using majority voting. Different from bagging, boosting subset creation is not random but depends upon the performance of the previous models, and every new subsets contain the misclassified instances by previous models.

We take decision-tree based models for both bagging and boosting, which is a very common approach. In the case of bagging, we consider a Bagging Tree model. We take an AdaBoost [8] Tree model for boosting, which uses decision trees as first level learners. AdaBoost (short for Adaptive Boosting) trains subsequent models in favor of those instances misclassified by previous ones. AdaBoost is sensitive to noisy data and outliers, but in general, it can be less susceptible to over-fitting.

3.8 Stacking

While bagging and boosting generally use the same type of model in all the different training steps (e.g., decision trees), stacking [9] aims at exploring the input data space through *base models* of different type. Stacking is the ensemble learning model that really makes use of a meta learner, which uses the output of the base learners as input for prediction. The point of stacking is to explore a space through the different properties of different models, each of them capable to learn some part of the problem, but not the whole space. The meta leaner is said to be stacked on the top of the other based models, hence the name.

General ensemble learning approaches might be prone to overfitting the data. In [1] a simple stacking learning algorithm named *Super Learner* is proposed as a possible solution for this overfitting limitation. It proposes a method to minimize the overfitting likelihood using a variant of cross-validation. In addition, the Super Learner provides performance bounds, as it performs asymptotically as good as the best available single hypothesis predictor, for each predicted pattern.

In the study, we consider two flavors of Super Learner for stacking, using the aforementioned single models as base learners: a simple majority voting based algorithm (Stacking MV), where the output of the base learners are equally weighted to decide on the final output, and GML (Generic Machine Learning), which basically computes weights in an exponential fashion, using the classification accuracy of each base learner. This approach permits to reduce the influence of low accuracy base predictors.

4 SCENARIOS AND DATASETS

In this section we overview the three network measurement problems we consider for evaluation. These include: (i) detection of network attacks [3], (ii) detection of smartphone-apps anomalies [4] and (iii) QoE prediction in cellular networks [20].

4.1 Detection of Network Attacks

The first problem consists of the analysis of diverse types of network attacks in real network traffic measurements collected at the WIDE backbone network, using the well-known MAWILab dataset for attacks labeling [12]. MAWILab is a public collection of 15-minute network traffic traces captured every day on a backbone link between Japan and the US since 2001. Building on this repository, the MAWILab project uses a combination of four traditional anomaly detectors (PCA, KL, Hough, and Gamma, see [12]) to partially label the collected traffic.

Table 1: Input features for detection of attacks.

Field	Feature	Description
Tot. volume	# pkts	num. packets
	# bytes	num. bytes
PKT size	pkt_h	H(PKT)
	pkt_{min,avg,max,std}	min/max/std, $\overline{\text{PKT}}$
	pkt_p{1,2,5,...95,97,99}	percentiles
IP Proto	# ip_protocols	num. diff. IP protocols
	ipp_h	H(IPP)
	ipp_{min,avg,max,std}	min/max/std, $\overline{\text{IPP}}$
	ipp_p{1,2,5,...95,97,99}	percentiles
	% icmp/tcp/udp	share of IP protocols
IP TTL	pkt_h	H(TTL)
	ttl_{min,avg,max,std}	min/max/std, $\overline{\text{TTL}}$
	ttl_p{1,2,5,...95,97,99}	percentiles
IPv4/IPv6	% IPv4/IPv6	share of IPv4/IPv6 pkts.
	# IP_src/dst	num. unique IPs
	top_ip_src/dst	most used IPs
TCP/UDP ports	# port_src/dst	num. unique ports
	top_port_src/dst	most used ports
	port_h	H(PORT)
	port_{min,avg,max,std}	min/max/std, $\overline{\text{PORT}}$
	port_p{1,2,5,...95,97,99}	percentiles
TCP flags (byte)	flags_h	H(TCPF)
	flags_{min,avg,max,std}	min/max/std, $\overline{\text{TCPF}}$
	flags_p{1,2,5,...95,97,99}	percentiles
	% SYN/ACK/PSH/...	share of TCP flags
TCP WIN size	win_h	H(WIN)
	win_{min,avg,max,std}	min/max/std, $\overline{\text{TCPF}}$
	win_p{1,2,5,...95,97,99}	percentiles

The traffic studied in this paper spans two months of packet traces collected in late 2015. From the labeled anomalies and attacks, we focus on a specific group which are detected simultaneously by the four MAWILab detectors, using in particular those events which are labeled as "anomalous" by MAWILab. We consider in particular 5 types of attacks/anomalies: (i) DDoS attacks (DDoS), (2) HTTP flashcrowds (mptp-la), (3) Flooding attacks (Ping flood), and two different flavors of distributed network scans (netscan) using (4) UDP and (5) TCP-ACK probing traffic. We train ML models to detect each of these attack types separately, thus each detection approach consists of five different detectors which run in parallel on top of the data, each of them specialized in detecting one of the five aforementioned attacks types. As a result, each detection approach can not only detect the occurrence of an attack, but also classify its nature.

To detect different attacks, we consider a slotted, time-based evaluation. For doing so, we split the traffic traces in consecutive time slots of one second each, and compute a set of features describing the traffic in each of these slots. In addition, each slot i is assigned a label l_i, consisting of a binary vector $l_i \in \mathbb{R}^{5\times 1}$ which indicates at each position if anomaly of type $j = 1..5$ is present or

Table 2: Input features for anomaly detection.

Field	Feature	Description
DNS_query	querycnt	# DNS requests
APN	apn_h	H(APN)
	apn_avg	$\overline{\text{APN}}$
	apn_p{99,75,50,25,05}	percentiles
Error_flag	error_code_h	H(Error_flag)
	error_code_avg	$\overline{\text{Error\_flag}}$
	error_code_p{99,75,50,25,05}	percentiles
Manufacturer	manufacturer_h	H(Manufacturer)
	manufacturer_avg	$\overline{\text{Manufacturer}}$
	manufacturer_p{99,75,50,25,05}	percentiles
OS	os_h	H(OS)
	os_avg	$\overline{\text{OS}}$
	os_p{99,75,50,25,05}	percentiles
FQDN	req_fqdn_h	H(FQDN)
	req_fqdn_avg	$\overline{\text{FQDN}}$
	req_fqdn_p{99,75,50,25,05}	percentiles

not in current time slot. We compute a large number n of features describing a time slot, using traditional packet measurements including traffic throughput, packet sizes, IP addresses and ports, transport protocols, flags, etc. Tab. 2 describes the set of n=245 features, which are computed for every time slot $i = 1..m$. Note that besides using traditional features such as min/avg/max values of some of the input measurements, we also consider the empirical distribution of some of them, sampling the empirical distribution at many different percentiles. This provides as input much richer information, as the complete distribution is taken into account. We also compute the empirical entropy $H(\cdot)$ of these distributions, reflecting the dispersion of the samples in the corresponding time slot.

4.2 Detection of Apps Anomalies

In [4] we conceived a semi-synthetic dataset for traffic anomalies in cellular networks by using real DNS traffic measurements. After collecting DNS traces for longer than six months in 2014 at a cellular network of a large-scale European operator, we devised a technique to generate new traffic traces by carefully recombining real traffic traces. Basically, we take samples of manually labeled one-minute intervals from the original data, characterized by a vector of features containing the distribution of DNS query counts by device Manufacturer, device OS, APN, domain name (FQDN) and DNS transaction flag. With the anomaly-free intervals we generate new synthetic background traffic, simply by shuffling the data samples of the same time of the day and same day class (working or festivity). Then, three different types of anomalies are introduced into the synthetic data, derived from real anomalies observed in this operational network. These anomalies mimic different types of service outages, and are represented by impacting a different number of end-users requesting particular services on specific domain names. The different anomalies considered are E1: short lived (hours) high intensity anomalies (e.g., 10% of devices repeating a request every few seconds), where the involved devices share the same manufacturer and OS; E2: several days lasting low intensity anomalies (e.g., 2% of devices repeating requests every few minutes) and E3: short-lived variable intensity anomalies affecting all devices of a specific APN. The used dataset consists of a full month of synthetically generated measurements, reported with a time granularity of 10 minutes time bins. Each time bin is assigned a class, either normal (label 0) or anomalous (label 1, 2 or 3 for the three anomaly types respectively). The dataset includes 16 different variations of E1, E2 and E3 anomalies, impacting a different fraction of end-users - going from 0.5% to 20%. Full details on the synthetic dataset are available in [4].

We take as main traffic feature the total number of DNS requests issued within a time bin. As we saw in [4], perturbations in this feature indicate that a device sub-population deviates from the usual DNS traffic patterns, thus pointing to potential anomalies. To better detect and diagnose the anomalies, we additionally take the distributions of DNS query counts across the aforementioned fields (Manufacturer, OS, APN, FQDN and DNS flag). From these distributions, we compute a set of features describing their shape and carried information, such as various percentiles and entropy values. Tab. 2 describes the specific set of n=36 features, which are computed for every time bin. The set includes the number of observed DNS requests, as well as multiple percentiles of fields such as associated APN, device OS and manufacturer, requested FQDN and number of DNS error messages. We also take as input the average values of these fields, as well as their entropy, the latter reflecting the dispersion of the observed samples.

Table 3: Input features for QoE prediction.

KPI Name	KPI Description (U – reported by user)
MOS	overall user experience **(U)**
ISP	cellular network operator
RAT	radio access technology
SIG	avg. signal strength
TH_{max}	max. session downlink flow throughput
TH_{avg}	avg. session downlink flow throughput
DUR	session duration
VOL	session volume
$\text{FLOW}_{\text{ratio}}$	ratio (# flows up)/(# flows down)
CELL	cell id
LOC	user location context **(U)**

4.3 Cellular QoE Prediction

For the sake of QoE prediction in cellular traffic, we use network and QoE measurements collected in a user field trial taking place in 2015 and detailed in [20], where 30 users equipped with their own devices connected to their preferred cellular operators evaluated three apps as part of their normal daily Internet activity during two weeks: YouTube (watching short videos); Facebook (timeline and photo-album browsing), and Gmaps (satellite maps browsing). QoE feedback was reported for each session through a customized QoE crowdsourcing app, according to a discrete, 5-levels ACR Mean Opinion Score (MOS) scale, ranging from "bad" (i.e., MOS = 1) to "excellent" (i.e., MOS = 5). In addition, each device has a passive flow-level traffic monitor which

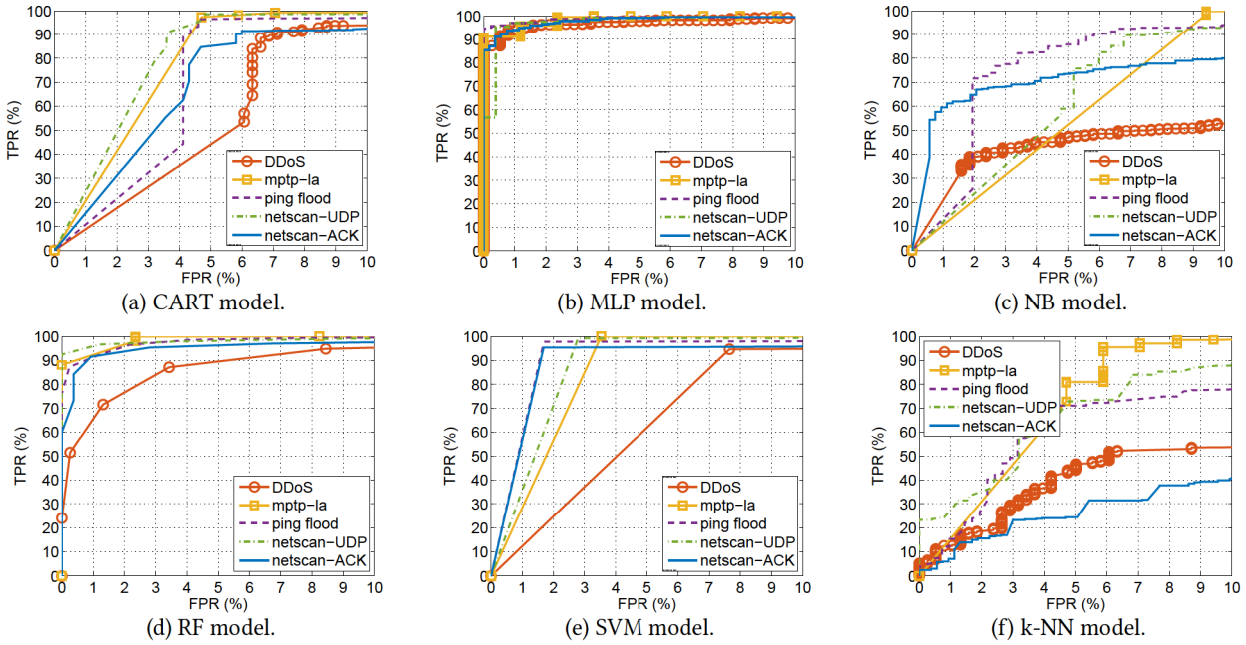


Figure 1: Detection performance (ROC curves) achieved by the base ML models for detection of network attacks.

records flow-level network traffic statistics, associating flows to apps generating them. The 10 different session-based KPIs in Tab. 3 are derived from the flow-based measurements, which are then synchronized to the QoE feedbacks (MOS scores) using time stamps. The KPIs include features such as average and maximum flow throughput per session, flow size, duration, average signal strength, RAT, ISP, locations, etc. The prediction problem consists in predicting the correct MOS score value (5-classes classification problem), using the session-based KPIs as input. Full details on the dataset are available in [20].

5 EVALUATION AND DISCUSSION

We now evaluate and compare the performance achieved by the presented ML models. To limit biased results, presented results correspond to 10-fold cross validation. All models are tested using scikit-learn python implementations. Parameters on each different algorithm are calibrated based on standard grid-based search tests. In addition, classes in each classification problem are balanced by statistical bootstrapping [20] to avoid unbalanced training issues. We start by comparing the performance achieved by the base learning models, and then present a full comparison including also the ensemble-learning approaches.

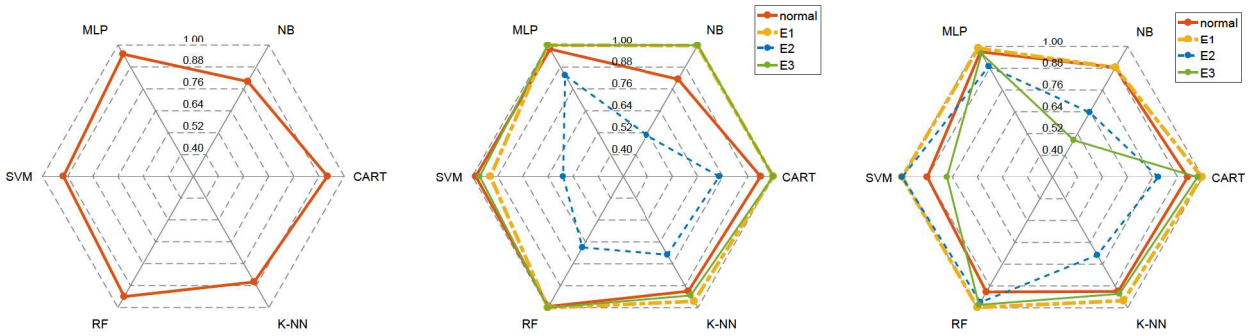
5.1 Single Base Learning Models

We start the analysis for the case of network attacks detection. Detection performance is measured by computing the True and False Positive Rates (TPR/FPR) for each model and for each of the attack types, using as input the full set of 245 features. Fig. 1 depicts the Receiver Operating Characteristic (ROC) curves obtained with each model, for the proposed attack classes. Besides the NB and the k-NN models, the tested approaches provide all highly accurate results for the five types of attacks. In general, detection performance is worse for DDoS attacks for all the evaluated models, suggesting that its fingerprint in the considered set of features is less marked than for the other attacks. Both the MLP and the RF models achieve the best performance, detecting around 80% of the attacks without false alarms.

Figs. 2 and 3 report the detection (prediction) performance achieved by the six base learning models for the case of anomaly detection and QoE prediction respectively. In both cases, detection is done along with classification (i.e., multi-class problems), thus we study the performance of each detector considering all the classes together. Performance is measured in terms of global classification accuracy (i.e., correctly classified instances), as well as per class recall and precision. Similar to the network attacks problem, Fig. 2 shows that both MLP and RF models are the most accurate ones for the sake of anomaly detection, even if some of the anomalies are more challenging to be correctly detected, for all the models. Indeed, anomalies of type E2 are harder to detect, basically due to their long-lasting, low intensity nature. In terms of QoE prediction, Fig. 3 clearly shows that decision-tree based models, and in particular RF ones, represent by far the most accurate approach, for all the different quality levels.

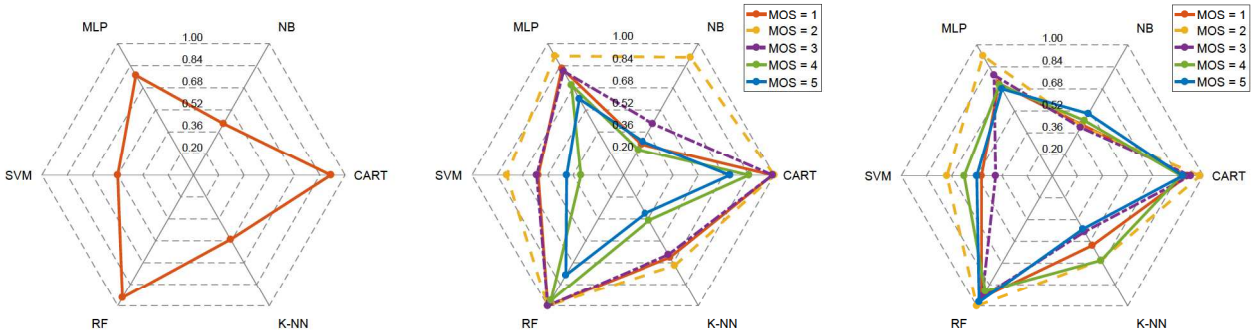
We focus now on the specific performance of the RF model for the three studied problems. To dig deeper into the RF outperformance, Fig. 4 depicts the ROC curves obtained with the RF model. We include in Fig. 4(a) the performance achieved by the RF model in the detection of network attacks for the sake of comparison to the other two problems. The first observation one could draw is that the RF model is much better for the anomaly detection and QoE prediction problems, but has a comparatively quite poor performance for detection of network attacks, i.e., when comparing at the problems/use-cases level. This point out to the first hypothesis we did in the introduction, suggesting that it is challenging to find a single model to properly tackle different problems simultaneously.

In the specific case of anomaly detection, Fig. 4(b) confirms that while anomalies of type E1 and E3 are perfectly detected by the RF model, anomalies of type E2 are quite often mis-classified as normal operation traffic. Fig. 4(c) shows that the RF model is



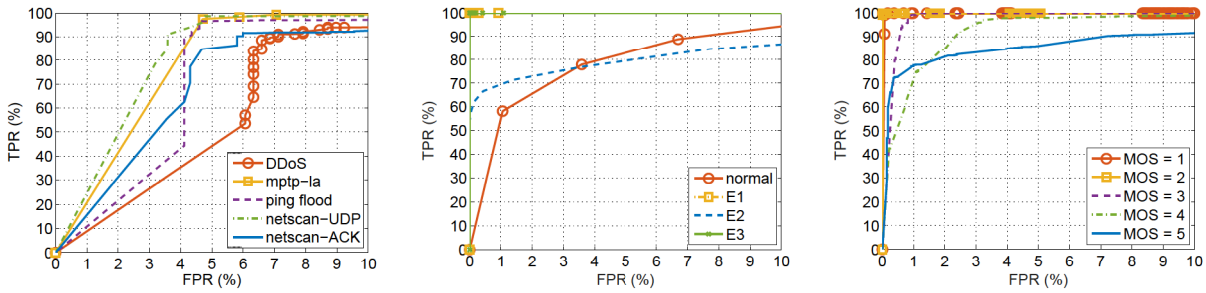
(a) Global Accuracy. (b) Recall. (c) Precision.

Figure 2: Global accuracy, recall and precision achieved by the base ML models for anomaly detection.



(a) Global Accuracy. (b) Recall. (c) Precision.

Figure 3: Global accuracy, recall and precision achieved by the base ML models for QoE prediction.



(a) Detection of Network Attacks. (b) Anomaly Detection. (c) QoE Prediction.

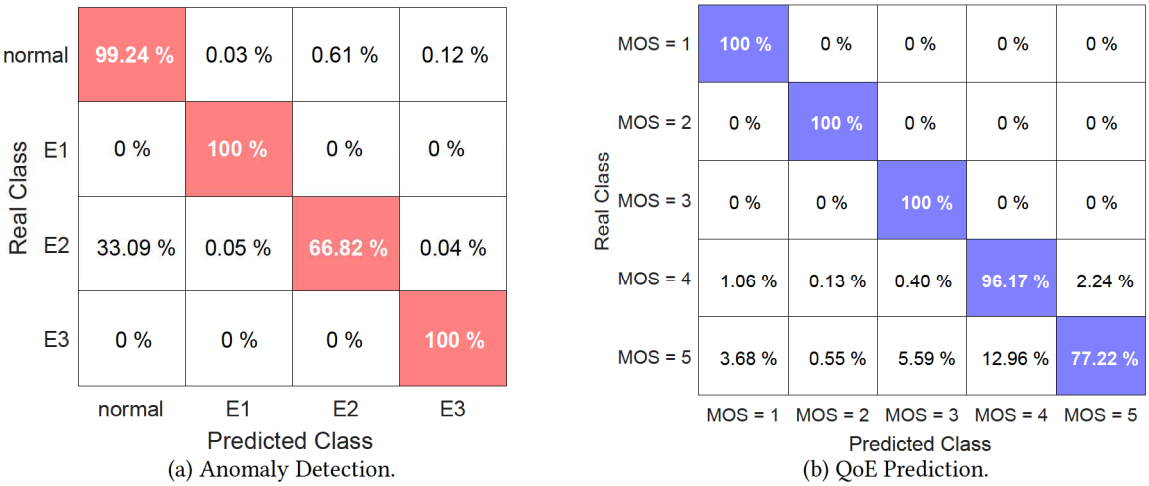
Figure 4: Performance of RF models - comparative ROC curves.

very accurate to correctly spot out bad quality sessions (i.e., MOS = 1, 2 and 3), but is less accurate to correctly predict higher quality ones. To better understand these mis-classification issues, Fig. 5 reports the corresponding confusion matrices obtained with the RF model in both problems. As reported in Fig. 5(a), about one third of the E2 anomalies go completely undetected within the normal traffic. In addition, as depicted in Fig. 5(b), excellent quality sessions (i.e., MOS = 5) are often misclassified as good ones (MOS = 4), and as average ones (i.e., MOS = 3) to a lesser extent.

5.2 Including Ensemble Learning Models

To conclude with the study, we now include the ensemble learning models within the analysis. Tabs. 4, 5 and 6 report the results obtained with all the models in the three problems, using the Area Under the ROC Curve (AUC) as performance metric. The machine learning community most often uses the ROC AUC statistic for model comparison [5], which is simple and informative.

Tab. 4 depicts the results obtained in the detection of the network attacks. The performance achieved by the ensemble learning models is generally higher than that of the single base learners alone. However, given that the single models performance is



(a) Anomaly Detection. (b) QoE Prediction.

Figure 5: Performance of RF models - Confusion matrices.

Table 4: ROC AUC for network attacks detection.

	DDoS	mptp-la	ping flood	scan-UDP	scan-ACK
CART	0.922	0.972	0.952	0.972	0.918
NB	0.828	0.952	0.963	0.944	0.929
MLP	0.982	0.995	0.997	**0.997**	0.989
SVM	0.935	0.982	0.980	0.982	0.968
RF	0.978	0.996	0.997	**0.997**	0.992
k-NN	0.762	0.964	0.890	0.936	0.843
Bagging Tree	**0.985**	0.995	0.995	0.996	0.988
AdaBoost Tree	0.940	0.997	0.996	0.995	0.983
Stacking MV	0.923	0.991	0.990	0.992	0.991
GML	**0.985**	**0.998**	**0.998**	**0.997**	**0.993**

Table 5: ROC AUC for anomaly detection.

	E1	E2	E3
CART	0.993	0.873	0.978
NB	0.956	0.861	0.959
MLP	0.997	0.944	0.996
SVM	0.996	0.944	0.995
RF	**0.999**	0.876	0.993
k-NN	0.995	0.859	0.963
Bagging Tree	0.996	0.885	0.983
AdaBoost Tree	0.998	0.945	0.995
Stacking MV	**0.999**	0.945	0.996
GML	**0.999**	**0.963**	**0.997**

already quite high, improvements are not that much significant. Boosting and bagging provide similar performance for all types of attacks, but it is stacking, and in particular the GML model, which provide the best results. Indeed, GML achieves the highest performance for all the five considered attack categories.

In the case of anomaly detection, as previously observed in Fig. 2, almost every predictor achieves an AUC over 99% for E1 anomalies. Thus, there is little room for improvement, which leads to only very subtle differences between the performances of base and ensemble-learners. Still, stacking learning models

Table 6: ROC AUC for QoE prediction.

MOS	1	2	3	4	5
CART	0.972	0.974	0.963	0.972	0.900
NB	0.766	0.874	0.714	0.707	0.703
MLP	0.916	0.951	0.918	0.852	0.798
SVM	0.812	0.928	0.742	0.717	0.734
RF	**0.992**	0.989	0.987	0.988	0.960
k-NN	0.849	0.917	0.765	0.756	0.657
Bagging Tree	0.971	0.977	0.996	0.982	0.955
AdaBoost Tree	0.972	0.978	**0.997**	0.992	0.973
Stacking MV	**0.992**	0.965	0.984	0.990	0.971
GML	**0.992**	**0.996**	**0.997**	**0.995**	**0.985**

tends to outperform both first level learners, as well as the bagging and boosting trees. Similar observations can be drawn from the detection of E3 anomalies. Note that the GML model systematically achieves the best results. For E2 anomalies, not only all predictors performed relatively poor, but also many of them achieve very low performance; e.g. the bagging models achieve an AUC below 90%, clearly worse than any other ensemble technique. This scenario highlights the advantages of the stacking models, and in particular, the GML model.

Finally, in the case of QoE prediction, Tab. 6 reveals again that predicting excellent QoE sessions (i.e., MOS = 5) is more challenging than for the rest of the quality levels. The CART and random forest models alone provide already very good results, as also shown in Fig. 3. Still, similar to the anomaly detection scenario, the GML model is capable to boost the prediction of excellent QoE w.r.t. both CART and random forest by at least 5% to 10%, suggesting again a better fit for this scenario.

While it is true that in some cases there is not enough room for improvement with respect to bagging, boosting and base learning models, the GML stacking model systematically outperforms these models for most of the tests and in all the problems, which opens the door for generalization of a technique for network measurement analysis. Indeed, stacking is less widely used than

bagging and boosting, but has recently shown outstanding performance in model competitions such as the Netflix Prize [22] and Kaggle competitions (https://www.kaggle.com/). The performance increase as compared to other ensemble learners is small, but in this case the computational overhead is similar, thus the GML model looks more appealing.

6 CONCLUDING REMARKS

In this paper we have demonstrated the outstanding performance of neural networks and decision trees for the analysis of network measurements coming from multiple and assorted networking problems. Based on the performance benchmarking, we can observe that both neural networks and decision-tree-based models provide in general better results in terms of accuracy and prediction than other single models, with a much smaller computational overhead for decision trees as compared to models based on neural networks or support vector machines. Decision-tree based models represent therefore a very appealing machine learning model for network analytics, not only because of their high accuracy and low computational cost, but also due to a series of embedded properties, such as model visibility, robustness to input noise, etc.

When looking at more complex models based on ensembles, we have shown the advantages of ensemble learning techniques to improve detection and prediction accuracy, and particularly of the stacking GML approach. We found that not only the GML based model has the ability to perform as well as the best input single base level learner, but often achieve better results. This includes also the case of both bagging and boosting models, which are also generally outperformed by the stacking models. Performance improvements are higher in scenarios where the performance of the base predictors is relatively low; when first learners performance is already high, there is little room for improvement. We believe that this study would enable a broader application of machine learning models to network analytics problems, with very promising results.

ACKNOWLEDGMENTS

The research leading to these results has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "Big-DAMA". Visit https://bigdama.ait.ac.at/ for details on the Big-DAMA project: Big Data Analytics for Network Traffic Monitoring and Analysis.

REFERENCES

[1] M. Van der Laan, et al., "Super learner", in Statistical applications in genetics and molecular biology, vol. 6, no. 1, 2007.
[2] M. Wang, et al., "Machine Learning for Networking: Workflow, Advances and Opportunities", in IEEE Network, issue 99, pp. 1 – 8, 2017.
[3] P. Casas, et al., "POSTER:(Semi)-Supervised Machine Learning Approaches for Network Security in High-Dimensional Network Data", in *ACM CCS*, 2016.
[4] P. Casas, et al., "Machine-learning based approaches for anomaly detection and classification in cellular networks", in *TMA*, 2016.
[5] A. J. Hanley, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases", Radiology, vol 148(3), pp. 839–843, 2008.
[6] L. Breiman, "Bagging Predictors", Machine Learning, vol. 24(2), pp. 123-140, 1996.
[7] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm", in *ICML*, 1996.
[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Journal of Computer and System Sciences, vol. 55(1), pp. 119-139, 1997.
[9] D. Wolpert, "Stacked Generalization", Neural Networks, vol. 5(2), pp. 241-259, 1992.
[10] D. Kwon, et al., "A Survey of Deep Learning-based Network Anomaly Detection", in Cluster Computing, The Journal of Networks, Software Tools and Applications, https://doi.org/10.1007/s10586-017-1117-8, 2017.
[11] T. Ahmed, et al., "Machine Learning Approaches to Network Anomaly Detection", in *USENIX SYSML Workshop*, 2007.
[12] R. Fontugne et al., "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking", *CoNEXT*, 2010.
[13] M. H. Bhuyan, et al., "Network Anomaly Detection: Methods, Systems and Tools", IEEE Communications Surveys & Tutorials, vol. 16 (1), pp. 303–336, 2014.
[14] M. Ahmed, et al., "A Survey of Network Anomaly Detection Techniques", J. Netw. Comput. Appl., vol. 60, pp. 19–31, 2016.
[15] A. L. Buczak, et al., "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection", IEEE Communications Surveys & Tutorials, vol. 18 (2), pp. 1153–1176, 2008.
[16] R. Ravinder, et al., "Real Time Anomaly Detection Using Ensembles", in *ICISA International Conference*, 2014.
[17] M. Ozdemir, I. Sogukpinar, "An Android Malware Detection Architecture based on Ensemble Learning", in Transactions on Machine Learning and Artificial Intelligence, vol. 2, no. 3, pp. 90–106, 2014.
[18] T. Nguyen et al., "A Survey of Techniques for Internet Traffic Classification using Machine Learning", IEEE Communications Surveys & Tutorials, vol. 10 (4), pp. 56-76, 2008.
[19] A. Balachandran, et al., "Developing a Predictive Model of Quality of Experience for Internet Video", *ACM SIGCOMM*, 2013.
[20] P. Casas et al., "Predicting QoE in Cellular Networks using Machine Learning and in-Smartphone Measurements", *QoMEX*, 2017.
[21] J. Vanerio et al., "Ensemble-learning Approaches for Network Security and Anomaly Detection," in *ACM SIGCOMM Big-DAMA workshop*, 2017.
[22] A. Töscher et al., "The BigChaos Solution to the Netflix Grand Prize," [on-line] http://www.netflixprize.com/