GML Learning, A Generic Machine Learning Model for Network Measurements Analysis

Pedro Casas
AIT Austrian Institute of Technology
pedro.casas@ait.ac.at

Juan Vanerio
Universidad de la República
jvanerio@fing.edu.uy

Kensuke Fukuda
National Institute of Informatics
kensuke@nii.ac.jp

***Abstract*—The application of machine learning 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 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 to address the analysis of large volumes of highly-dimensional data collected in operational networks. In this paper we present a potential solution to fill this gap, exploring the application of ensemble learning models to multiple network measurement problems. We introduce GML Learning, a generic Machine Learning model for the analysis of network measurements. The GML model is a generalization of the well-known stacking approach to ensemble learning, and follows the concepts of the Super Learner model. The Super Learner performs asymptotically as well as the best input base or weak learners, providing a very powerful approach to tackle multiple problems with the same technique. In addition, it defines an approach to minimize over-fitting likelihood during training, using a variant of cross-validation. We deploy the GML model on top of Big-DAMA, a big data analytics framework for network measurement applications. We test the proposed solution in five different and assorted network measurement problems, including detection of network attacks and anomalies, QoE modeling and prediction, and Internet-paths dynamics tracking. Results confirm that the GML model provides better results than any of the single baseline models of the stack, and outperforms traditional bagging and boosting ensemble learning approaches. The GML Learning model opens the door for a generalization of a best-practice technique for the analysis of network measurements.**

***Index Terms*—Big-Data; Network Traffic Monitoring and Analysis; Machine Learning; Ensemble Learning; Super Learning; Network Measurements.**

I. Introduction

Data-driven networking [1], 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 are however two major challenges in applying machine learning models at large-scale for handling network measurements: (i) from a practical perspective, network measurement applications require to process in near real-time very large amounts of fast and heterogeneous network monitoring data. Network monitoring data usually comes in the form of high-speed streams, which need to be rapidly and continuously analyzed. Different systems have been conceived in the past to collect large amounts of measurements in operational networks, but a flexible data processing system capable to analyze and extract useful insights from such rich data is needed; there is also a need to process massive amounts of network measurement data in an off-line fashion with better and more elaborated data analytic models, which is not trivial, specially when applying machine learning models, that have in general heavy-computational requirements; (ii) from a theoretical perspective, 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. The ensemble learning theory permits to combine multiple single models to form a (hopefully) better one. 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. In addition, an ensemble of models exhibits higher robustness with respect to uncertainties in training data.

In this paper we tackle both challenges through the application of big data analytics and big data platforms. To address the practical challenge, we overview Big-DAMA, a Big Data Analytics Framework (BDAF) for network monitoring applications, designed with comprehensive network monitoring in mind. Starting from a predecessor system called DBStream [26], [27], we have conceived a flexible BDAF, capable of analyzing and storing big amounts of both structured and unstructured heterogeneous data sources, with both stream and batch processing capabilities. Big-DAMA implements multiple data analytics algorithms for network measurements analysis, using both supervised and unsupervised machine learning models. These models are implemented using off-the-shelf machine learning libraries.

The research leading to these results has been partially funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "BigDAMA".

To address the theoretical challenge, we devise GML Learning, a novel data analytics model for the analysis of network measurements using the Super Learner ensemble learning model [2]. The Super Learner is a loss-based ensemble-learning method that finds the optimal combination of a collection of base prediction algorithms. The Super Learner performs asymptotically as well as the best input base or *weak* learner, providing a very powerful approach to tackle multiple problems with the same technique. In addition, it defines an approach to minimize over-fitting likelihood during training, using a variant of cross-validation. Ensemble learning has in principle a higher computational cost and complexity than single based learning approaches, as multiple models have to be trained and applied to the analyzed data. However, the usage of Big-DAMA permits to alleviate this constraint by allowing the parallel execution of multiple machine learning models.

To show the performance and generalization of GML Learning as well as its application on top of Big-DAMA, we use GML learning on the analysis of five different and assorted network measurement problems, including detection of apps anomalies, detection of network attacks, QoE prediction, QoE-modeling for video streaming, and Internet path-dynamics tracking. The presented evaluations confirm that the proposed GML learning model has the ability to perform as well as the best available base learning model, achieving even better results in most problems. We also show that the model improves analysis results as compared to other traditional ensemble learning models such as bagging and boosting.

We believe that this study would enable a broader application of ensemble learning models, and in particular of GML Learning, to data-driven networking problems. This paper builds on top of our recent early work on ensemble-learning models [43], where we explore the application of ensemble-learning techniques to network security and anomaly detection. In particular, we extend [43] by deploying and testing the proposed algorithms on top of the Big-DAMA big data analytics platform, by analyzing new network measurement problems, as well as by adding other ensemble-learning approaches based on bagging and boosting into the comparisons.

The reminder is organized as follows. Sec. II presents an overview on the related work on BDAFs and machine-learning based network measurement. In Sec. III we describe the main characteristics of the Big-DAMA BDAF. Sec. IV describes the main concepts behind the GML learning paradigm and introduces the evaluated models, developed within Big-DAMA. Sec. V describes the tested network measurement problems and datasets. Sec. VI presents the experimental results of the study, applying Big-DAMA to the analysis of these problems. Section VII concludes the paper.

II. State of the Art

This paper deals with machine learning and big data platforms for the analysis of network measurements, thus we slightly overview both domains, with an emphasis on big data platforms for network monitoring. 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 [12], machine learning for network traffic classification [18] and network security [15], machine learning models for QoE modeling [20] and prediction [21], as well as machine learning for Internet path performance analysis [24], [25]. While the application of learning techniques to network measurement problems is largely extended in the literature, 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].

Regarding big data processing systems, the big data boom of recent years has led to a very strong and fast development of novel solutions [29]. An overview of past and current Big Data Analysis Frameworks includes traditional Database Management Systems (DBMS) and extended Data Stream Management Systems (DSMSs), NoSQL systems, and Graph-oriented systems. While most target the off-line analysis of static data, more recent systems target the on-line analysis of data streams. DSMSs such as Gigascope [30] and Borealis [31] support continuous on-line processing, but cannot run off-line analytics over static data. The Data Stream Warehousing (DSW) paradigm can handle both on-line and off-line processing requirements within a single system. DataCell, DataDepot [28] and DBStream [26], [27] are examples of DSWs. NoSQL systems such as MapReduce [32] have also rapidly evolved, supporting the analysis of unstructured data. Apache Hadoop [33] and Spark [34] are very popular implementations of MapReduce systems. These are based on off-line processing rather than stream processing. Besides these systems, there is a large range of alternatives, including Hive, Hawq, Greenplum (SQL-oriented); Giraph, GraphLab, Pregel (graph-oriented), as well as well-known DBMSs commercial solutions such as Teradata, Dataupia, Vertica and Oracle Exadata (just to name a few of them). There has been promising recent work on enabling real-time analytics in NoSQL systems, such as Spark Streaming [35], Indoop [36], Muppet [37], SCALLA [38], as well as Storm, Samza and Flink, but most of them remain unexploited in the network monitoring domain.

BDAFs based on Hadoop have been proposed within the network monitoring domain [39]–[42]. However, the main drawback of such systems in general is their inherent off-line processing, which is not suitable for real-time traffic analysis.

III. The Big-DAMA Framework

The main purpose of Big-DAMA is to analyze and store large amounts of network monitoring data. Big-DAMA uses off-the-shelf big data storage and processing engines to offer both stream and batch processing capabilities, following a standard lambda architecture, decomposing separate frameworks for stream, batch and query. Lambda architecture is a data processing architecture designed to handle massive quantities of data by taking advantage of both batch- and stream-processing methods. This approach to architecture attempts

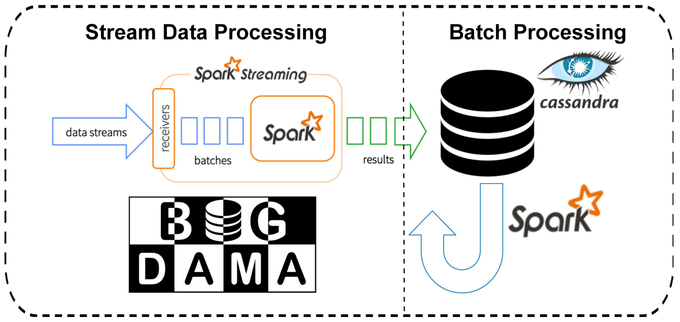

Figure 1: Data Stream Warehouse-based architecture for Big-DAMA platform, using Hadoop ecosystem.

to balance latency, throughput, and fault tolerance by using batch processing to provide comprehensive and accurate views of batch data, while simultaneously using real-time stream processing to provide on-line data analysis capabilities.

In a nutshell, Big-DAMA uses Apache Spark streaming for stream-based analysis, Spark for batch analysis, and Apache Cassandra for query and storage. There are two main reasons for using Cassandra instead of simply HDFS or Hadoop-based DBs such as Hive: fault-tolerance and speed. Cassandra is fully distributed and has no single point of failure, whereas HDFS has a single point-of-failure represented by the HDFS name nodes. Regarding speed, Cassandra has been built from scratch for the particular case of on-line transactional data, whereas HDFS follows a more static data warehousing perspective. In addition, Cassandra is highly scalable and provides linear scalability without compromising processing performance. Finally, being a NoSQL system it allows to store and handle multiple sources of heterogeneous data, including unstructured data.

Fig. 1 shows a high level architectural design of Big-DAMA. Inspired on DBStream [26], [27], the Big-DAMA BDAF follows a DSW paradigm, offering the possibility of combining on-the-fly data processing with large-scale storage and analytic capabilities. This paradigm provides the means to handle both types of on-line and off-line processing requirements within a single system.

Within the Big-DAMA BDAF, we have conceived different algorithms for the analysis of network measurements using supervised and unsupervised machine learning models. These models are currently implemented on top of the Big-DAMA batch-processing branch, using off-the-shelf, Spark ML machine learning libraries. Big-DAMA is currently deployed on top of a virtualized data cluster, consisting of 12 virtual nodes with a total capacity of 150 GB of memory and 30 TB of data storage. Big data frameworks are partially managed through a Cloudera, Hadoop ecosystem installation (https://www.cloudera.com/), using distribution CDH 5.10 and Cloudera Manager (with Spark 2).

IV. GML Learning in Big-DAMA

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.

Rather than finding the best model to explain the data, combined methods construct a set of models and then decide between them with some combinatorial approach, seeking complementarity in the sense that the learning limitations of each predictor compensates for the others. Thus, the execution of several of these algorithms in parallel provides diversity of predictions. Several research papers have studied methods exploiting this diversity to enhance the overall prediction capability by combining the outputs of multiple algorithms [5], [6]. Essentially, this is made by using a general scheme known as *Ensemble Learning*. There are multiple approaches to ensemble learning, including bagging [8], boosting [9], and stacking [11]. All three are so-called “meta-algorithms”, defining different approaches to combine machine learning techniques into one single model referred to as *meta learner*, to either decrease variance (bagging), decrease bias (boosting) or improve predictive performance (stacking).

Bagging - for Bootstrap Aggregation, 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. The well known Random Forest model is an example of bagging ensemble learning.

Boosting 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.

While bagging and boosting generally use the same type of model in all the different training steps (e.g., Random Forest), stacking aims at exploring the input data space through *base models* of different type. Stacking is the ensemble learning model which 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. Stacking is less widely used than bagging and boosting, but has recently shown outstanding performance in model competitions such as the Netflix Prize [44] and Kaggle competitions (https://www.kaggle.com/).

General ensemble learning approaches might be prone to over-fitting the data. In [2] a simple stacking learning algorithm named *Super Learner* is proposed as a possible solution for this over-fitting limitation. It proposes a method to minimize the over-fitting likelihood using a variant of cross-validation. In addition, the Super Learner provides performance bounds, as it performs asymptotically as good as the best single base predictor.

The Super Learner algorithm makes aggressive use of cross validation: the available labeled dataset consisting of n samples is split in K approximately equal sets. As usual, each of these sets is used as a *validation set*, while its complement, $K-1$ sets are used as the *training set*. For each split, the J first level learners are fitted with the training dataset and then do predictions for the samples of the validation set. By merging the predictions done for every fold we obtain a new dataset Z of size $n \times J$, containing the predictions done by each first level learner for every sample in the disjoint validation sets. This new dataset Z is used as design input matrix to train the meta learner algorithm, which will then be used to perform the final predictions. In the original paper [2], the meta learner can be arbitrarily complex, yet a simple linear regression model is used for the presented regression scenario. The paper presents a formal proof showing that this Super Learner is optimal in the sense that it can perform at least asymptotically as well as the best first level learner available. The performance measure must be a certain loss function that allows for risk calculation.

The logic expressed in [2] can be adapted for use on classification problems such as the one we tackle in this paper. Essentially, suppose there are n i.i.d. observations $(X_i, y_i) \sim P_0$ with $i = 1, \ldots, n$ that generate empirical probability distributions P_n, and that the goal is to estimate the classification function ψ_0 such that:

$$\psi_0(X) = \underset{\psi \in \Psi}{\arg\min}\ E\left[L\left(y, \psi(X)\right)\right] \tag{1}$$

where $L\left(y, \psi(X)\right)$ is a given loss function that measures the discrepancy between prediction and real value - e.g., square loss in [2], for all possible feature vector $X \in \mathcal{X}$ and its corresponding label $y \in \mathcal{Y}$. ψ_0 is then a mapping function from the feature space into the label space and Ψ the parameter space of all possible functions such that $\mathcal{X} \to \mathcal{Y}$. Now let $\{\hat{\psi}_j\}$ $j = 1, \ldots, J$ be the collection of first level learners, which represent mappings from the empirical distribution P_n into parameter space Ψ.

When using K-fold cross-validation let $k \in \{1, \ldots, K\}$ be the index of a split of the data into a validation set $V(k)$ and its complement, the training set $T(k)$. Let then $k(i)$ be the split index in which sample i belongs to the validation set, i.e. $i \in V(k(i))$ and $f_{j,T(k)}$ the realization of the j^{th}-first level learner $\hat{\psi}_j$ after being trained in $T(k)$ - assuming the training has as target the minimization of the expected risk $E[L(y, \hat{\psi}_j(X))]$. Then, a new observations dataset $Z = \{(Z_i, y_i)\}$ is constructed such that the i^{th}-sample $z_i = \left\{f_{j,T(k(i))} : j = 1, \ldots, J\right\}$ is the vector of the predictions of the J first level learners for sample i when sample i is not in the training dataset.

The last input for the Super Learner algorithm is another user defined algorithm $\phi\colon \{\mathcal{Y}\}^J \to \mathcal{Y}$, that shall be used as a predictor for labels $y \in \mathcal{Y}$ from data points $z \in \{\mathcal{Y}\}^J$. This algorithm must also be trained to minimize the expected risk in a similar fashion to the first level learners, that is to become similar to the optimal mapping:

$$\phi^*(Z) = \underset{\phi \in \Phi}{\arg\min} E\left[L\left(Y, \phi(Z)\right)\right] \tag{2}$$

over the set Φ of functions $\{\mathcal{Y}\}^J \to \mathcal{Y}$. Although not the case presented in [2], this fitting can be made using penalization or cross-validation to further avoid over-fitting. Let then $g\colon \{\mathcal{Y}\}^J \to \mathcal{Y}$ be the function obtained from fitting algorithm ϕ with training dataset $\{Z_i\}$ and label set $\{y_i\}$.

Once g has been determined, the first level learners are re-trained on the whole available training dataset to obtain the fitted predictors $\{f_j : j = 1, \ldots, J\}$. Thus, the Super Learner algorithm becomes a new algorithm S such that:

$$S(X_i) = g\left(f_1(X_i), \ldots, f_J(X_i)\right) \tag{3}$$

The GML learning model is a particular implementation of the Super Learner, using a probability-based weighting function to combine the outputs of the first level learners. In a nutshell, we use the probabilities of success of each class to build exponentially decayed weighting functions, adding a control variable to reduce the overall influence of low accuracy models in the final prediction.

As a final note, the outputs of the learning models can be categorical in case of a *hard decision* or a score in a *soft decision* case. The latter is more expressive, as it provides an extra degree of freedom in the selection of the decision threshold and allows for performance descriptions such as *Receiver Operation Characteristic (ROC)* curves. Thus, we prefer to use as output from each algorithm the probability of the evaluated sample belonging to the corresponding classes; as such, the elements of matrix Z represent probabilities.

A. First Level Learners

Ensembles of machine learning models tend to yield better results when there is a significant diversity among the individual base models. Therefore, we select an assorted group of base learning models with very different underlying data assumptions. In particular, we select the following five standard, fully-supervised models [18]: (i) SVM with linear kernel - linear kernel is used to improve speed w.r.t. default RBF kernels, (ii) decision trees (CART); (iii) K-NN with direct majority voting, using $K = 10$, (iv) multi layer perceptron neural network (MLP), and (v) Naïve Bayes (NB). Most of these models have already shown good performance in previous work on network security, anomaly detection and classification, and QoE prediction [3], [4], [21]. The hyper-parameter configuration values for each model are selected on a manual basis, by trial and error as well as by following default recommended settings. All models are implemented on top of Big-DAMA, using python and off-the-shelf ML libraries.

B. Super Learners and GML

The original work [2] uses a simple minimum square linear regression as the example Super Learner. Following the Super Learner logic described before, we conceived five different Super Learner algorithms, one of them being the GML learning model. As we are dealing with classification problems, a first natural choice is the usage of logistic regression, which shall be the first evaluated Super Learner.

In [6], a linear weighted algorithm is suggested as meta-learner for ensemble learning, by taking predictions from each first-level learner and weighting them to get a wighted-majority-voting-like classifier; more concrete, let $H(X) = \sum_{j=1}^{J} w_j h_j(X)$ be the weighted sum of the individual first-level learner predictions $h_h(X)$, the algorithm decides for the positive class if $H(X) > \beta$, being β the decision threshold, or the negative class otherwise. The weights w_j can be defined in different ways; in this work we use three different types of weights:

MVuniform: gives the same weight $(1/J)$ to each learner, implementing simple majority voting.

MVaccuracy: assigns weights $w_j = \dfrac{\alpha_j}{\sum_{i=1}^{J} \alpha_i}$ to the prediction of learner j, being α_i the *probability of success* of the learner - i.e., the fraction of true classifications achieved on the whole available training dataset.

GML: the GML Learner computes weights with an exponential classification probability, $w_j = \dfrac{e^{\lambda \alpha_j}}{\sum_{i=1}^{J} e^{\lambda \alpha_i}}$, where λ is selected to reduce the influence of low accuracy predictors - we take $\lambda = 10$ for such an effect. As we show next, this property of reducing the influence of poor first level models makes a significant difference in the final results.

Finally, [2] mentions that there is no need to restrict the Super Learner algorithm to parametric regression or classification fits. For example, one could define it in terms of a particular machine learning algorithm. To also test this direction, we devise another Super Learner based on a simple decision tree model, using the well known CART decision tree algorithm.

C. Bagging and Boosting Algorithms

We take decision-tree based models for both bagging and boosting, which is a very common approach. In the case of bagging, we consider two different flavors of the same algorithm: a Bagging Tree model, and a Random Forest [18]. The main difference is that in Random Forest, only a subset of features are selected at random out of the total for split at each node, reducing correlation between trees. In bagging, all features are considered for splitting a node. To have comparable results, these models use as many internal decision trees as first level learners has the Super Learner; i.e., 5 in this paper.

We take an AdaBoost [10] 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.

V. Measurement Problems and Datasets

In this section we briefly overview the five network measurement problems we take for evaluation. These include: (i) detection of smartphone-apps anomalies [4], detection of network attacks [3], QoE prediction in cellular networks [21], QoE-modeling for video streaming [23], and Internet-paths dynamics tracking [25].

A. 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 app-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. From the aforementioned distributions, we compute a set of 36 features describing their shape and information, such as various percentiles and entropy values, which are computed for every time bin. 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].

B. Detection of Network Attacks

The detection of network attacks is done on top of MAWI data [19]. MAWI is a public collection of 15-minute real 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 to partially label the collected traffic [19]. From the labeled anomalies and attacks, we focus on a specific group which are detected simultaneously as "anomalous" by the four MAWILab detectors to achieve a high quality on the obtained labels. We consider five types of attacks/anomalies in particular: (i) DDoS attacks (DDoS), (ii) HTTP flashcrowds (mptp-la), (iii) Flooding Attacks (Ping Flood), (iv) UDP and (v) TCP probing traffic. The dataset spans a full week of MAWILab traffic traces collected in late 2015; traces are split in consecutive time slots of one second each, and a high-dimensional set of 245 features describing the traffic in each of these slots is used, see [3] for more details.

C. 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 [21], 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 [22], 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 records flow-level network traffic statistics, associating flows to apps generating them. 10 different session-based KPIs 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 [21].

D. Video QoE Modeling

For the video streaming QoE modeling problem, we use a publicly available subjective QoE measurements dataset released in 2016. The LIVE-Avvasi Mobile Video database [23] consists of 174 distorted videos generated from 24 reference videos with 26 unique stalling events and 4830 ratings obtained from 54 subjects who viewed the videos on mobile devices. Reference videos correspond to HD content from YouTube and Vimeo, with a duration range between 29 and 134 seconds (after adding stalling events). Video content spans different categories, including more dynamic contents such as sports to more stable contents such as documentaries, as well as advertisement and music clips. Quality ratings are provided on a standard single stimulus, continuous scale basis, but reported as a Degradation MOS score (DMOS), using also a 5-levels scale. From the base dataset, we extract 19 different input features for each video session, characterizing the stalling patterns undergone by the videos, as well as the particular video contents. Features focus on number and frequency of stalling events, initial playback delay, duration of stallings, as well as their particular location within the video stream.

E. Prediction of Internet Path Changes

The last problem we consider for analysis consists of predicting the number of changes an Internet path experiences during a period of 24 hours. For doing so, we use standard Paris traceroute measurements, performed through the M-Lab open Internet measurement initiative (https://www.measurementlab.net/) . The M-Lab infrastructure consists of a high number of servers distributed globally in multiple provider networks and geographic regions, mostly in the US. The raw traceroute data files are publicly available through Google's BigQuery and Cloud Storage, see https://console.cloud.google.com/storage/browser/m-lab/. For the purpose of this case study, we analyze the first week in January 2016 of traceroute measurements, which correspond to more than 450,000 different paths, measured from more than 180 geo-distributed servers.

We predict the number of path changes in a certain day using features collected during the previous day; these features include 69 input metrics describing the statistical properties of the dynamics and latency of a path. Full details on the dataset are available in [25]. To turn the problem into a classification one, we build three non-overlapping classes accounting for the number of daily path changes as follows: (i) *static paths*, corresponding to 0 path changes; (ii) *dynamic paths*, corresponding to 1-to-10 path changes; and (iii) *very dynamic paths*, corresponding to more than 10 path changes.

VI. Evaluation and Discussion

In this section we show that the GML Learner approach can enhance the results obtained on the proposed problems by outperforming both first level learners (CART, SVM, MLP, kNN and NB) as well as other Super Learners and ensemble models. We compare the performance achieved by each single, first-level model to that achieved by the four proposed Super Learners (logistic regression, MVuniform, MVaccuracy and CART-based), the two bagging models (Bagging Tree and Random Forest), the AdaBoost tree learning model, and finally, the GML learning model.

Comparisons are conducted using Big-DAMA and Spark ML like pipe-lines in python. To limit biased results, presented results correspond to 10-fold cross validation. As performance metric, we take the Area Under the ROC Curve (AUC) for each of the corresponding classes. The machine learning community most often uses the ROC AUC statistic for model comparison [7]. Parameters on each different algorithm are calibrated based on trial and error search tests. In addition, classes in each classification problem are balanced by statistical bootstrapping [21] to avoid unbalanced training issues.

Table I: ROC AUC for anomaly detection.

	E1	E2	E3
CART	0.993	0.873	0.978
Naïve Bayes	0.956	0.861	0.959
MLP	0.997	0.944	0.996
SVM	0.996	0.944	0.995
kNN	0.995	0.859	0.963
Random Forest	**0.999**	0.876	0.993
Bagging Tree	0.996	0.885	0.983
AdaBoost Tree	0.998	0.945	0.995
logreg	**0.999**	0.952	0.996
MVaccuracy	**0.999**	0.948	0.996
MVuniform	**0.999**	0.945	0.996
CART	0.997	0.924	0.994
GML	**0.999**	**0.963**	**0.997**

Table II: ROC AUC for network attacks detection.

	DDoS	HTTP	S-TCP	S-UDP	Flooding
CART	0.745	0.856	0.909	0.923	0.928
Naïve Bayes	0.730	0.655	0.897	0.933	0.917
MLP	0.907	0.993	0.979	0.983	0.989
SVM	0.883	0.992	0.941	0.995	0.968
kNN	0.720	0.936	0.936	0.924	0.944
Random Forest	0.827	0.905	0.941	0.913	0.930
Bagging Tree	0.823	0.908	0.911	0.915	0.921
AdaBoost Tree	0.892	0.991	0.923	0.920	0.927
logreg	0.926	0.956	0.952	0.980	0.987
MVaccuracy	0.924	0.992	0.971	0.993	**0.993**
MVuniform	0.923	0.991	0.970	0.992	0.991
CART	0.867	0.992	0.933	0.985	0.954
GML	**0.935**	**0.998**	**0.983**	**0.997**	**0.993**

Tab. I reports the results obtained in the detection of apps anomalies, using the corresponding AUC values. Achieved results for anomalies of type E1 and E3 differs completely from E2. 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 Super Learners, base learners and GML. Still, the GML learning model 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 Super Learner models, and in particular, the GML model.

Tab. II depicts the results obtained in the detection of the network attacks. In this scenario the differences w.r.t. GML are more relevant, specially for the detection of DDoS attacks, which are poorly detected by first level models alone, but better with Super Learners (e.g., MVaccuracy and MVuniform) and with GML. Again, when first level model performance is high, there is little room for improvement with GML, but still, the proposed model achieves better performance for the five considered attack categories.

Table III: ROC AUC for QoE prediction.

MOS	1	2	3	4	5
CART	0.972	0.974	0.963	0.972	0.900
Naïve Bayes	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
kNN	0.849	0.917	0.765	0.756	0.657
Random Forest	**0.992**	0.989	0.987	0.988	0.960
Bagging Tree	0.971	0.977	0.996	0.982	0.955
AdaBoost Tree	0.972	0.978	0.997	0.992	0.973
logreg	0.970	**0.996**	0.983	0.991	0.969
MVaccuracy	0.980	0.972	**0.997**	0.987	**0.985**
MVuniform	**0.992**	0.965	0.984	0.990	0.971
CART	0.975	0.964	0.974	0.980	0.891
GML	**0.992**	**0.996**	**0.997**	**0.995**	**0.985**

Table IV: ROC AUC for QoE modeling and assessment.

DMOS	1	2	3	4	5
CART	0.977	0.973	0.970	0.955	0.888
Naïve Bayes	0.784	0.881	0.711	0.706	0.692
MLP	0.893	0.949	0.921	0.850	0.798
SVM	0.821	0.927	0.734	0.720	0.749
kNN	0.852	0.932	0.774	0.759	0.649
Random Forest	0.979	**0.995**	0.976	0.979	0.964
Bagging Tree	0.967	0.991	0.985	0.982	0.953
AdaBoost Tree	0.975	0.985	0.989	0.990	0.984
logreg	**0.997**	0.984	0.954	0.997	0.958
MVaccuracy	**0.997**	0.979	**0.998**	0.996	0.985
MVuniform	0.979	0.981	0.987	**0.998**	0.977
CART	0.996	0.969	0.966	0.971	0.902
GML	**0.997**	0.993	**0.998**	0.996	**0.999**

Table V: ROC AUC for path dynamics prediction.

	static	dynamic	very dynamic
CART	0.967	0.964	0.969
Naïve Bayes	0.950	0.883	0.939
MLP	0.961	0.956	**0.977**
SVM	0.951	0.833	0.875
kNN	0.932	0.897	0.931
Random Forest	**0.989**	0.957	0.963
Bagging Tree	0.971	0.972	**0.977**
AdaBoost Tree	0.988	0.791	0.847
logreg	0.957	0.896	0.970
MVaccuracy	0.981	0.975	0.962
MVuniform	0.983	0.965	**0.977**
CART	0.965	0.974	0.965
GML	**0.989**	**0.979**	**0.977**

Tab. III presents the results obtained in the prediction of QoE MOS scores in cellular services. Note first that predicting excellent QoE sessions (i.e., MOS = 5) is more challenging than for the rest of the quality levels. A deeper analysis of the classification confusion matrix reveals that MOS = 5 class is often misclassified as MOS = 4, showing that it's difficult to distinguish between excellent and good QoE using network layer metrics only. The CART and random forest models alone provide already very good results, as also shown previously in [21]. The GML model is capable to boost the prediction of excellent QoE w.r.t. both CART and random forest by 9% and 3% respectively, thus improving results obtained in [21].

Table VI: ROC AUC average values for the five measurement problems.

	AD	NS	QoE-P	QoE-M	PPC	ALL
CART	0.948 (**3.9%**)	0.872 (**11.1%**)	0.956 (**3.7%**)	0.952 (**4.4%**)	0.966 (**1.9%**)	0.935 (**5.4%**)
Naïve Bayes	0.925 (**6.2%**)	0.826 (**15.8%**)	0.752 (**24.2%**)	0.754 (**24.3%**)	0.924 (**6.3%**)	0.819 (**17.1%**)
MLP	0.979 (**0.7%**)	0.970 (**1.1%**)	0.887 (**10.7%**)	0.882 (**11.5%**)	0.964 (**2.1%**)	0.929 (**6.0%**)
SVM	0.978 (**0.8%**)	0.955 (**2.6%**)	0.786 (**20.8%**)	0.790 (**20.7%**)	0.886 (**10.1%**)	0.869 (**12.1%**)
kNN	0.939 (**4.8%**)	0.892 (**9.1%**)	0.788 (**20.6%**)	0.793 (**20.4%**)	0.920 (**6.7%**)	0.854 (**13.6%**)
Random Forest	0.956 (**3.1%**)	0.903 (**7.9%**)	0.983 (**1%**)	0.978 (**1.8%**)	0.969 (**1.6%**)	0.957 (**3.2%**)
Bagging Tree	0.954 (**3.2%**)	0.895 (**8.7%**)	0.976 (**1.7%**)	0.975 (**2.1%**)	0.973 (**1.3%**)	0.953 (**3.6%**)
AdaBoost Tree	0.979 (**0.7%**)	0.930 (**5.2%**)	0.982 (**1.1%**)	0.984 (**1.2%**)	0.875 (**11.2%**)	0.954 (**3.5%**)
logreg	0.982 (**0.4%**)	0.960 (**2.1%**)	0.981 (**1.1%**)	0.978 (**1.9%**)	0.941 (**4.5%**)	0.970 (**1.9%**)
MVaccuracy	0.981 (**0.5%**)	0.974 (**0.7%**)	0.984 (**0.9%**)	0.991 (**0.6%**)	0.972 (**1.3%**)	0.981 (**0.8%**)
MVuniform	0.980 (**0.6%**)	0.973 (**0.8%**)	0.980 (**1.3%**)	0.984 (**1.2%**)	0.980 (**0.5%**)	0.979 (**1.0%**)
CART	0.971 (**1.5%**)	0.946 (**3.6%**)	0.956 (**3.6%**)	0.960 (**3.6%**)	0.968 (**1.8%**)	0.959 (**3.0%**)
GML	**0.986**	**0.981**	**0.993**	**0.996**	**0.985**	**0.989**

Tab. IV reports the results obtained in the modeling of QoE for video streaming in smartphone devices. Even if achieving excellent results for the five corresponding classes, the GML learning model is outperformed by the random forest model in the prediction of the DMOS = 2 class, corresponding to perceived annoying quality distortion. Still, the GML model is ranked second for this class, and outperforms all the other ensemble-learning and first level models.

Finally, Tab. V presents the results obtained in the prediction of Internet path changes. Predicting static paths is easier for all the tested models. Some first level models such as SVM fail to properly predict changes in dynamic and highly dynamic paths, and also boosting fails to achieve good results for the two classes. The GML model performs again better than most of the tested models for the three classes, obtaining the same performance as random forest, neural networks and both bagging tree and super learning in some of the classes, but ranking first in each category.

To conclude with the comparative analysis, Tab. VI summarizes the performance achieved by all the models for the five tested problems, considering the average values for all the classes of each problem - Anomaly Detection (AD), Network Security (NS), QoE Prediction (QoE-P), QoE-Modeling (QoE-M) and Prediction of Path Changes (PPC). The last column of the table provides the average AUC values for all problems together. Next to each AUC value, there is a fraction indicating the performance increase provided by GML, which permits to have a full view on the applicability and goodness of the GML model on the five network measurement problems studies in this paper. The GML model outperforms all other models in the five problems, with performance increases ranging from as low as 0.5% for some problems and some Super Learners, up to about 20% and 25% for first level models such as SVM and Naïves Bayes in the QoE-related problems. Looking at the average performance increase for ALL the problems, there is clear distinction between first level models - performance increase ranges from 5.4% to 17.1%, bagging and boosting - performance increase of approximately 3%, and Super Learners - performance increase from 0.8% up to 3%. While it is true that in some scenarios there is not enough room for improvement w.r.t. bagging, boosting and first level models, and thus the additional computational cost of GML might not be justified, the GML model systematically outperforms these models for most of the tests and in all the problems, which also opens the door for generalization of a technique for network measurement analysis. The performance increase w.r.t. other Super Learners is small, but in this case the computational overhead is exactly the same, thus the GML model is more appealing. Last but not least, as we said before, the usage of high-performance big data platforms such as Big-DAMA permits to eliminate the problems introduced by extra computational times, by allowing the parallel execution of multiple models. The benchmarking of computational times of all the tested algorithms running on top of Big-DAMA is part of our ongoing work.

VII. Concluding Remarks

In this paper, we have presented GML learning, a generic Machine Learning model for the analysis of network measurements. We have demonstrated the advantages of GML for the analysis of network measurements coming from multiple and assorted networking problems; GML does not only have the ability to outperform the most accurate first level learner, but also outperforms other ensemble-learning models based on bagging, boosting and stacking. We have also described Big-DAMA, a big data analytics framework specially tailored for network monitoring applications. The performance improvements of GML are higher in scenarios where the performance of the first level predictors were relatively low; when first learners performance is already high, there is little room for improvement. The different evaluated Super Learner schemes achieved very similar performances. However, the GML model performs the best for all scenarios, suggesting a potentially good approach to go for by default in similar classification problems. The simplicity and very low computational costs of GML majority voting scheme makes a very nice case for such type of models. We believe that this study would enable a broader application of big data analytics and big data platforms to network measurement problems, with very promising results.

References

[1] J. Jiang, et al., "Unleashing the Potential of Data-Driven Networking", in *COMSNETS*, 2017.

[2] M. Van der Laan, et al., "Super learner", in Statistical applications in genetics and molecular biology, vol. 6, no. 1, 2007.

[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] Y. Freund, et al., "Using and Combining Predictors that Specialize", in *ACM STOC*, 1997.

[6] T. Dietterich, "Ensemble learning", The handbook of brain theory and neural networks, vol. 2, pp. 110–125, MIT Press: Cambridge, MA, 2002.

[7] 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.

[8] L. Breiman, "Bagging Predictors", Machine Learning, vol. 24(2), pp. 123-140, 1996.

[9] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm", in *ICML*, 1996.

[10] 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.

[11] D. Wolpert, "Stacked Generalization", Neural Networks, vol. 5(2), pp. 241-259, 1992.

[12] T. Ahmed, et al., "Machine Learning Approaches to Network Anomaly Detection", in *USENIX SYSML Workshop*, 2007.

[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] R. Fontugne et al., "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking", *CoNEXT*, 2010.

[20] A. Balachandran, et al., "Developing a Predictive Model of Quality of Experience for Internet Video", *ACM SIGCOMM*, 2013.

[21] P. Casas et al., "Predicting QoE in Cellular Networks using Machine Learning and in-Smartphone Measurements", *QoMEX*, 2017.

[22] Int. Telecommunication Union, "ITU-T Rec. P.800: Methods for Subjective Determination of Transmission Quality," 1996.

[23] D. Ghadiyaram et al., "LIVE Mobile Stall Video Database", [online]: http://live.ece.utexas.edu/research/LIVEStallStudy/index.html, 2016.

[24] I. Cunha et al., "DTRACK: A System to Predict and Track Internet Path Changes", IEEE/ACM Transactions on Networking, vol. 22 (4), pp. 1025–1038, 2014.

[25] S. Wassermann et al., "NETPerfTrace - Predicting Internet Path Dynamics and Performance with Machine Learning", *ACM SIGCOMM Big-DAMA*, 2017.

[26] A. Bär et al., "Large-Scale Network Traffic Monitoring with DBStream, a System for Rolling Big Data Analysis," *IEEE Big Data*, 2014.

[27] A. Bär et al., "DBStream: an Online Aggregation, Filtering and Processing System for Network Traffic Monitoring", *TRAC*, 2014.

[28] L. Golab et al., "Stream Warehousing with DataDepot," *SIGMOD*, 2009.

[29] M. Stonebraker, "SQL Databases vs. NoSQL Databases," *Comm. of the ACM*, vol. 53(4), pp. 10-11, 2010.

[30] C. Cranor et al., "Gigascope: A Stream Database for Network Applications," *SIGMOD*, 2003.

[31] D. Abadi et al., "Aurora: A New Model and Architecture for Data Stream Management," *The VLDB Journal*, 12(2), pp. 1020-1039, 2003.

[32] J. Dean et al., "MapReduce: Simplified Data Processing on Large Clusters," *Comm. of the ACM*, 51(1), pp. 107-113, 2008.

[33] T. White, "Hadoop: the Definitive Guide," *O'Reilly Media, Inc.*, 2009.

[34] M. Zaharia et al., "Spark: Cluster Computing with Working Sets," *HotCloud*'10.

[35] M. Zaharia, et al., "Discretized Streams: An Efficient and Fault-tolerant Model for Stream Processing on Large Clusters," *HotCloud*, 2012.

[36] P. Bhatotia et al., "Indoop: Mapreduce for Incremental Computations," *SoCC*'11.

[37] W. Lam et al., "Muppet: Mapreduce-style processing of fast data," *Proc. VLDB Endow.*, vol. 5(12), pp.1814-1825, 2012.

[38] B. Li et al., "Scalla: A platform for scalable one-pass analytics using mapreduce," *ACM Trans. Database Syst.* 37(4), pp. 27-43, 2012.

[39] R. Fontugne et al., "Hashdoop: A MapReduce Framework for Network Anomaly Detection," *IEEE INFOCOM Workshops*, 2014.

[40] Y. Lee et al., "Toward scalable internet traffic measurement and analysis with Hadoop," in *SIGCOMM Comput. Commun. Rev.*, 43(1), pp. 5-13, 2012.

[41] J. Liu et al., "Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop," *IEEE Network*, 28(4), pp. 32-39, 2014.

[42] M. Wullink et al., "ENTRADA: a High-Performance Network Traffic Data Streaming Warehouse," *IEEE/IFIP NOMS*, 2016.

[43] J. Vanerio et al., "Ensemble-learning Approaches for Network Security and Anomaly Detection," in *ACM SIGCOMM Big-DAMA workshop*, 2017.

[44] A. Töscher et al., "The BigChaos Solution to the Netflix Grand Prize," [on-line] http://www.netflixprize.com/