Super Learning for Anomaly Detection in Cellular Networks

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

Juan Vanerio
AIT Austrian Institute of Technology & Universidad de la República
email: jvanerio@fing.edu.uy

***Abstract*—The ever-growing population of smartphones connected to mobile networks is changing the cellular traffic ecosystem. The traffic volumes and patterns generated by smartphone apps pose complex challenges to cellular network operators, particularly in terms of detection and diagnosis of network anomalies caused by specific apps. The high-dimensionality of network data provided by current network monitoring systems opens the door to the application of machine learning approaches to improve the detection and classification of network anomalies, but this higher dimensionality comes with an extra data processing overhead. In addition, critical network monitoring applications such as the detection of anomalies require fast mechanisms for on-line analysis of thousands of events per second, as well as efficient techniques for off-line analysis of massive historical data. In this paper we explore the application of big data analytics and big data platforms to the automatic detection of anomalies in mobile networks. We consider ensemble, multi-combined machine learning models to enhance anomaly detection, following a particularly promising model known as Super Learning. Super-learning is an ensemble learning approach which performs asymptotically as well as the best possible weighted combination of multiple learning algorithms, providing a very powerful detection approach. We implement and test different super-learning models on top of Big-DAMA, a big data analytics framework for network monitoring. We test the proposed solution on the detection of traffic anomalies in operational cellular networks, and compare it to other traditional ensemble learning approaches such as bagging and boosting. Results indicate that our approach outperforms traditional ones.**

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

I. Introduction

Network anomaly detection represents a key component for network management. The high-volume and high-dimensionality of network data provided by current network monitoring systems opens the door to the application of machine learning approaches to improve the detection and classification of anomalous events.

There are however two major challenges in applying machine learning models at large-scale for network monitoring: (i) in a more practical perspective, network monitoring applications require to process in near real-time very large amounts of fast and heterogeneous network monitoring data, which is not trivial, specially when applying machine learning models, which have in general heavy-computational requirements. 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; (ii) in a more 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 for 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, which is highly beneficial.

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 [27], we have conceived a first running prototype of the Big-DAMA BDAF. Big-DAMA is 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 anomaly detection using both supervised and unsupervised machine learning models. These models are implemented using off-the-shelf machine learning libraries.

To address the theoretical challenge, we devise a novel detection technique for network anomaly detection using the Super Learner ensemble learning model [1]. The Super Learner is a loss-based ensemble-learning method that finds the optimal combination of a collection of base prediction

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

algorithms. The Super Learner performs asymptotically as well as the best possible weighted combination of the base 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. Ensemble learning has in principle a higher computational cost and complexity than single learning based 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 application of Big-DAMA and the performance of various Super Learning models in an operational monitoring application, we apply the Big-DAMA BDAF to the detection of different network traffic anomalies on top of a semi-synthetic dataset, statistically derived from network measurements collected in an operational cellular network. Evaluations confirm that super learning techniques have the ability to perform as well as the best available base learning model, achieving even better results in most scenarios and using different combination approaches. We also show that the super learner improves detection 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 super learning models to the detection of anomalies in network traffic. This paper builds on top of our recent early work on big-data analytics [26], where we present the basics of the Big-DAMA project, and on ensemble-learning models [44], where we explore the application of ensemble-learning techniques to network security and anomaly detection. In particular, we extend [44] by deploying and testing the detection algorithms on top of the Big-DAMA big data analytics platform, as well as by adding other ensemble-learning approaches based on bagging and boosting into the comparisons.

The remainder of the paper is organized as follows. Section II presents an overview on the related work on BDAFs and anomaly detection techniques. In Section III we describe the main characteristics of the Big-DAMA BDAF. Section IV describes the main concepts behind the super learning paradigm and introduces the evaluated models, developed within Big-DAMA. Section V presents the experimental results of the study, applying Big-DAMA and multiple super learners to the detection of anomalies in cellular network traffic. Finally, Section VI concludes the paper.

II. State of the Art

This paper deals with machine learning and big data platforms for network anomaly detection, thus we slightly overview both domains, with an emphasis on big data platforms for network monitoring. There are a couple of extensive surveys on generic-domain anomaly detection techniques [18] as well as network anomaly detection [19], [20], including machine learning-based approaches. The application of learning techniques to the problems of anomaly detection is largely extended in the literature. A large set of papers apply concepts and techniques imported from fields like neural networks [10], self-organizing maps [11], genetic algorithms [12], fuzzy logic [13], machine learning [14]–[17], etc.

The specific application of ensemble learning approaches to anomaly detection is by far more limited, and even if it is generally observed in the practice that ensembles tend to yield better results when there is a significant diversity among the models, only few papers have applied them to network anomaly detection [21], [22].

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 [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 recently 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 data processing 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 to balance latency, throughput, and fault tolerance by using batch processing to provide comprehensive

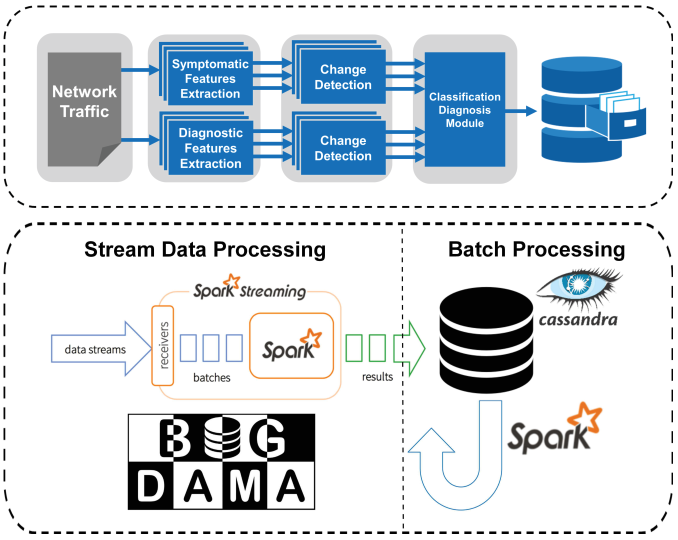

Figure 1: (top) Automatic Network Anomaly Detection and Diagnosis system - ANAD^2. (bottom) Data Stream Warehouse-based architecture for Big-DAMA platform, using Hadoop ecosystem. ANAD^2 runs on top of Big-DAMA.

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 [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.

For the purpose of anomaly detection, Big-DAMA implements the Automatic Network Anomaly Detection and Diagnosis system (ANAD^2) we previously conceived in [25]. Following this framework, network traffic is processed on the fly and two different types of features are extracted: symptomatic features and diagnostic features. All features are analyzed for the sake of anomaly detection. However, the symptomatic features are designed such that their changes directly relate to the presence of abnormal and potentially harmful events. On the other hand, changes in the diagnostic features per-se do not have a negative connotation, but rather ease and guide the interpretation of the anomalous event. Within the Big-DAMA BDAF, we have conceived different algorithms for network anomaly detection 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.

Following the recently introduced data flow model [43], we are currently exploring the adoption of Apache Beam, an advanced unified programming model, which simplifies the implementation of both batch and streaming data processing jobs which can run on the base Spark execution engine used by Big-DAMA, by offering a single unified programming model.

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, connected through Open vSwitch technology. The physical infrastructure consists of 3 physical server nodes, each equipped with 2 Intel Xeon(R) E5-2630 v2-2.60GHz CPUs (24 cores total) and with a total capacity of 64 GB of memory. Virtualization is achieved by Linux Kernel-based VMs, using Proxmox Virtual Environment (https://www.proxmox.com/) for management and orchestration. 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. Super 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.

According to [5], single hypothesis algorithms or simple learning models may suffer from three different bottlenecks: **statistical problem** - arises when the space of hypothesis is too large for the amount of available training data, resulting in several algorithms with similar accuracy and risk of choosing one that will not predict future data points well; **computational problem** - several algorithms are not guaranteed to find the global optimum; **representation problem** - results from the hypothesis space not containing any model that is a good approximations of the true distribution.

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 [4], [5]. Essentially, this is made by using a general scheme known as *Ensemble Learning*. There are multiple approaches to ensemble learning, including bagging [6], boosting [7], and stacking [9]. All three are so-called “meta-algorithms”, defining different approaches to combine several machine learning techniques into one single predictive model referred to as *meta learner*, to either decrease the variance (bagging), decrease the bias (boosting) or improve the 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 [45] and Kaggle competitions (https://www.kaggle.com/).

General ensemble learning approaches might be prone to over-fitting the data. In [1] 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 available single hypothesis predictor, for each predicted pattern.

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 [1], 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 [1] 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 [1], 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 $\mathcal{X} \rightarrow \mathcal{Y}$. 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 $\phi\colon \{\mathcal{Y}\}^J \rightarrow \mathcal{Y}$, 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:

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

over the set Φ of functions $\{\mathcal{Y}\}^J \rightarrow \mathcal{Y}$. Although not the case presented in [1], this fitting can be made using penalization or cross-validation to further avoid over-fitting. Let then $g \colon \{\mathcal{Y}\}^J \rightarrow \mathcal{Y}$ 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:

$$S(X_i) = g\left(f_1(X_i), \ldots, f_J(X_i)\right) \qquad (3)$$

As a final note, the outputs of the first level learners and the Super Learner 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 decided to use as output from each algorithm the probability of the evaluated sample belonging to the "positive" classes (i.e., detection of anomalies); as such, the elements of matrix Z represent probabilities and, similar to the most generic case for X, are also continuous values.

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 [24]: (i) SVM with linear kernel - liner 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, and (v) naive Bayes. Most of these models have already shown good performance in previous work on anomaly detection and classification [2], [3]. 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. These models are trained according to the previously described procedure, first to create the matrix Z, and then re-trained on the whole available training dataset to make predictions on the testing dataset. In the evaluations, we compare the individual performance of each of these base models to the performance achieved by the devised super learners algorithms, described next. All models are implemented on Big-DAMA, using python and off-the-shelf ML libraries.

B. Super Learner Algorithms

The original work [1] 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. 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 [5], 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 $w_j = \frac{\alpha_j}{\sum_{i=1}^{J} \alpha_i}$ to the prediction of learner j, being α_i the *accuracy* of the learner - i.e., the fraction of true classifications achieved on the whole available training dataset.

MVexp: computes weights with an exponential classification accuracy, $w_j = \frac{e^{\lambda \alpha_j}}{\sum_{i=1}^{J} e^{\lambda \alpha_i}}$, where λ is selected to reduce the influence of low accuracy predictors - we take $\lambda = 10$ for such an effect.

Finally, [1] 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. 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 [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.

Table I: Input features for anomaly detection.

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

V. Evaluation and Discussion

In this section we show that the Super Learner approach can enhance the results obtained on the classification problem studied in [3] for detection of network anomalies. In that study, different standard first-level learning models are used. An exception to this is the usage of Random Forests [24], which actually represents a bagging learning approach. The Random Forest algorithm is more sophisticated than the Super Learner itself, in the sense that it already performs feature selection during training.

We therefore compare the performance achieved by each single, first-level detector against that achieved by the proposed Super Learners. We also compare the performance of the Super Learners against that achieved by the AdaBoost Tree learning model, the Bagging Tree model, and the Random Forest.

Evaluations are performed on two different scenarios: firstly, we partially reproduce the results obtained in [3], by re-implementing some of the tested algorithms on Big-DAMA; then, we run a full comparison between base learners, super learners and the other ensemble learners, using Big-DAMA and Spark ML like pipe-lines in python. As performance figures and metrics, we take true positive vs false alarm rates (TPR/FPR respectively) through Receiver Operating Characteristic (ROC) curves and the corresponding Area Under the ROC Curve (AUC) values.

A. Data Description

To test the performance of the proposed approaches, we apply Big-DAMA on the analysis of a semi-synthetic dataset for traffic anomalies in cellular networks, conceived in [3] 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 [3].

We take as main traffic feature the total number of DNS requests issued within a time bin. As we saw in [3], 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. I 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.

For the sake of training and testing, we split the dataset in two, temporally-disjoint sub-sets: a training set, with the first week of traffic - approximately 20% of the samples, and a testing set with the remaining three and a half weeks - 80% of samples. While 10-fold cross-validation would provide better results, we take such a split to follow the super-learning approach, and to test a practical scenario, for which one would train the models on a first time period of collected measurements (e.g., 1 week) and apply the resulting system in the following time slots.

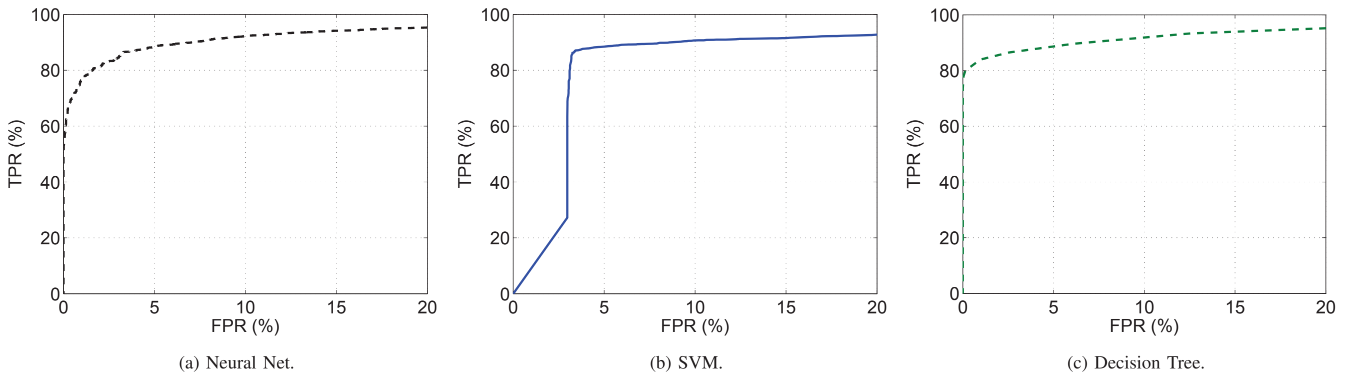

Figure 2: ROC curves showing anomaly detection performance for single-model detectors, for all anomaly types.

Table II: ROC AUC for the detection of different service anomalies in an operational cellular network.

	E1	E2	E3
Neural Net		0.971	
SVM		0.925	
Decision Tree		0.980	
Decision Tree	0.993	0.873	0.978
Naive Bayes	0.956	0.861	0.959
Neural Net	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
MVexp	**0.999**	**0.963**	**0.997**
MVuniforme	**0.999**	0.945	0.996
CART	0.997	0.924	0.994

B. *Detection of Network Anomalies with Single Models*

We first evaluate the implementation of three basic ML algorithms on top of Big-DAMA, partially reproducing the results obtained in [3]. These algorithms include neural networks, support vector machines, and decision trees. Fig. 2 depicts the ROC curves obtained for the detection of the complete set of anomalies (i.e., E1 + E2 + E3) using the three models. The first three rows of Tab. II report the corresponding AUC values.

The SVM model achieves slightly worse detection performance, resulting in a true positives rate close to 90% but with a false alarm rate above 3%. Both the neural network and the decision tree models achieve better performance, detecting around 60% and 80% of the anomalies respectively without false alarms, but they also fail to detect many anomalies. As we show next, the main issue is with anomalies of type E2, specially with those of lower intensity.

The obtained results are in line with the accuracy, precision and recall values presented in [3], which are implemented in a single-node platform and not in Big-DAMA.

C. *Super Learning Detection*

We now evaluate the performance of the proposed super learning models. We compare the performance of the first level learners as described in Sec. IV-A, as well as the performance of the super learners described in Sec. IV-B, and that of the other ensemble learners described in Sec. IV-C. Tab. II reports the results obtained by each detector and for each anomaly type, using the corresponding AUC values. Similar to [3], 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 and base learners. Still, super learning models yens to outperform both the 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 MVexp model systematically achieves the best results.

The attempt to detect E2 anomalies shows quite different results. 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 clearly highlights the advantages of the Super Learner approach for anomaly detection, as all the Super Learners, expect from CART, outperform the first level and the bagging and boosting learners.

VI. Concluding Remarks

In this paper, we claim that applying machine learning models at large-scale for network monitoring has become paramount for detecting anomalies in complex networks, but that there are both practical and theoretical associated challenges which need to be addressed first.

We have therefore described Big-DAMA, a big data analytics framework specially tailored for network monitoring applications. Using off-the-shelf big data storage and processing engines, Big-DAMA is capable of analyzing and storing big amounts of both structured and unstructured heterogeneous data sources, with both stream and batch processing capabilities. We have shown the types of ML-based algorithms implemented in Big-DAMA for network anomaly detection, using off-the-shelf ML libraries.

To solve the model selection problem and to improve detection accuracy, we have demonstrated the advantages of ensemble learning techniques for detection of anomalies, and particularly of the Super Learner approach. By benchmarking different models on the detection of traffic anomalies in operational cellular networks we found that not only the Super Learner based predictors have the ability to perform as well as the best available first level learner, but often achieve better results. This includes also the case of both bagging and boosting models, which are also outperformed by the super learning models.

The performance improvements 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 MVexp Super Learner performs the best for all anomaly types, suggesting a potentially good approach to go for by default in similar classification problems. The simplicity and very low computational costs of MVexp majority voting 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 anomaly detection, with very promising results.

References

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

[2] P. Casas, et al., "POSTER:(Semi)-Supervised Machine Learning Approaches for Network Security in High-Dimensional Network Data", in *ACM CCS*, 2016.

[3] P. Casas, et al., "Machine-learning based approaches for anomaly detection and classification in cellular networks", in *TMA*, 2016.

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

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

[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] A. Ghosh, A. Schwartzbard, "A Study in Using Neural Networks for Anomaly and Misuse Detection", in *USENIX Security Symposium*, 1999.

[11] A. Mitrokotsa et al., "Detecting denial of service attacks using emergent self-organizing maps", in *IEEE ISSPIT*, 2005.

[12] M. Ostaszewski et al., "A non-self space approach to network anomaly detection", in *IEEE IPDPS*, 2006.

[13] W. Chimphlee, et al., "Integrating genetic algorithms and fuzzy C-means for anomaly detection", in *IEEE Indicon*, 2005.

[14] G.Prashanth, et al., "Using random forests for network-based anomaly detection", in *IEEE ICSCN*, 2008.

[15] Y. Li et al., "An efficient network anomaly detection scheme based on TCM-KNN algorithm and data reduction mechanism", in *IAW*, 2007.

[16] P. Casas et al., "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge", in *Computer Communications*, vol. 35 (7), pp. 772-783, 2011.

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

[18] V. Chandola, et al., "Anomaly detection: A survey", ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, 2009.

[19] M. Ahmed, et al., "A Survey of Network Anomaly Detection Techniques", J. Netw. Comput. Appl., vol. 60, pp. 19–31, 2016.

[20] W. Zhang, et al., "A Survey of Anomaly Detection Methods in Networks", in *CNMT Symposium*, 2009.

[21] R. Ravinder, et al., "Real Time Anomaly Detection Using Ensembles", in *ICISA International Conference*, 2014.

[22] 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.

[23] V. Chandola et al., "Anomaly Detection: a Survey", *Com. Sur.*, vol. 41 (3), 2009.

[24] T. Nguyen et al., "A Survey of Techniques for Internet Traffic Classification using Machine Learning", *IEEE Comm, Surv. & Tut.*, vol. 10 (4), pp. 56-76, 2008.

[25] P. Fiadino et al., "RCATool - A Framework for Detecting and Diagnosing Anomalies in Cellular Networks", *ITC 27*, 2015.

[26] P. Casas et al., "Big-DAMA: Big Data Analytics for Network Traffic Monitoring and Analysis", *ACM SIGCOMM LANCOMM workshop*, 2016.

[27] A. Baer et al., "Large-Scale Network Traffic Monitoring with DBStream, a System for Rolling Big Data Analysis," *IEEE Big Data*, 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] T. Akidau et al., "The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," in *VLDB 41*, 2015.

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

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