Neurocomputing. Volume 343, 2019, Pages 100-119. https://doi.org/10.1016/j.neucom.2018.07.091

Exploratory Study on Class Imbalance and Solutions for Network Traffic Classification

Santiago Egea Gómez[a]*, Luis Hernández-Callejo[a], Belén Carro Martínez[a], Antonio J. Sánchez-Esguevillas[a]

[a] Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad de Valladolid, Campus Miguel Delibes, Valladolid 47011, Spain

ARTICLE INFO

Article history:

ABSTRACT

Network Traffic Classification is a fundamental component in network management, and the fast-paced advances in Machine Learning have motivated the application of learning techniques to identify network traffic. The intrinsic features of Internet networks lead to imbalanced class distributions when datasets are conformed, phenomena called Class Imbalance and that is attaching an increasing attention in many research fields. In spite of performance losses due to Class Imbalance, this issue has not been thoroughly studied in Network Traffic Classification and some previous works are limited to few solutions and/or assumed misleading methodological approaches. In this article, we deal with Class Imbalance in Network Traffic Classification, studying the presence of this phenomenon and analyzing a wide number of solutions in two different Internet environments: a lab network and a high-speed backbone. Namely, we experimented with 21 data-level algorithms, six ensemble methods and one cost-level approach. Throughout the experiments performed, we have applied the most recent methodological aspects for imbalanced problems, such as: DOB-SCV validation approach or the performance metrics assumed. And last but not least, the strategies to tune parameters and our algorithm implementations to adapt binary methods to multiclass problems are presented and shared with the research community, including two ensemble techniques used for the first time in Machine Learning to the best of our knowledge. Our experimental results reveal that some techniques mitigated Class Imbalance with interesting benefit for traffic classification models. More specifically, some algorithms reached increases greater than 8% in overall accuracy and greater than 4% in AUC-ROC for the most challenging network scenario.

Keywords: Machine Learning; Network management; Class Imbalance; Network Traffic Classification

1. Introduction

Internet network administrators often confront vast amounts of traffic and fast events happening in different points of Internet networks. Controlling and managing network resources can be an arduous task considering the fast increase of interconnected devices and the complexity of underlying network topologies. Due to the former facts, the provision of automatic tools to facilitate the network administrators' work is crucial and urgent. Network Traffic Classification (NTC) is a fundamental functionality of network management systems, since many cyber-attacks and network flaws can be easily detected via monitoring the network traffic. Thereby, researchers have shown an increasing interest in NTC recently [1].

Machine Learning (ML) has opened up promising future prospects for NTC and the number of published articles proposing traffic classifiers based on ML is increasing continuously [1]–[10]. The application of ML to NTC brings important advantages over previous approaches; however new challenges have risen up and they must be solved to accomplish feasible classifiers. Port-based classifiers [11] are the earliest and simplest techniques to characterize Internet traffic. This kind of classifiers relies on port numbers into IP headers to associate protocols and applications with flow connections according to the well-known ports defined by the IANA [12]. Unfortunately, emerging applications (predominantly peer-to-peer) that dynamically use different ports and/or deliberately mask their communications behind IANA ports impose an unresolved obstacle for port-based classifiers. This handicap motivated researchers to develop more sophisticated techniques, gaining a relevant relevance an approach known as Deep Packet Inspection (DPI). DPI tools [13] inspect binary information found in the application layer of network packets in order to seek matches between inspected packets and prefixed signatures. Although network hardware is fast evolving and, thus, the perspective of DPI tools are improving in some network scenarios, these techniques have major drawbacks to be implemented in network devices with scarce memory and computation resources. DPI approaches are pretty computationally weighted complicating their scalability, and additionally signature databases are quite difficult to maintain due to zero-day protocols and software updates. But the most limiting issue from the point of view of Internet Service Providers (ISPs) is users' privacy violation. DPI tools unceasingly extract information from the application layer accessing to personal information about network users. The above reasons are being motivating the advanced research on ML-based NTC, since ML essentially provides accurate and fast classifiers respecting users' privacy [1]–[3].

ML provides a wide number of preprocessing techniques and learning algorithms enabling highly accurate classifiers. Learning algorithms are able to process the knowledge contained in training datasets and generate predictive models describing the structure of data. The resulting models are afterwards used to reproduce the response for incoming unknown samples. If training datasets include the response to predict, we are solving a supervised learning task; otherwise, it is an unsupervised problem. Regarding the type of response, the modeling task is a classification problem if the response is categorical; whereas the regression problems cover cases in which the responses take continuous values.

NTC is a multiclass classification problem, since traffic classifiers aim to categorize objects (Internet connection flows) in different classes or traffic categories (protocols or applications). The most extended approach in ML-based NTC is flow-based level in which all packets associated with a connection are aggregated and jointly processed to create classification objects. Both

*Corresponding author.
E-mail address: santiago.egea@alumnos.uva.es

supervised and unsupervised approaches [14] have been proposed over recent years evidencing the potential of ML for NTC. Although unsupervised learning techniques have interesting advantages, such as the no necessity of a labeling process [1], supervised algorithms have outperformed unsupervised techniques in terms of accuracy. Furthermore, semi-supervised techniques [5] have also been studied with promising results. In this work we approach flow-based NTC from a supervised perspective.

Network environments impose important challenges when ML is employed. One of the main challenges is Class Imbalance, phenomena that is being actively studied in numerous research fields in which ML is applied [15] (such as: Banking Fraud [16], Computer Vision [17] and Medical Diagnosis [18]). A classification problem is categorized as imbalanced when one or various classes are overrepresented comparing to the others. In almost all network environments some services are more often consumed than others, which turns out non-uniform class distributions when NTC datasets are conformed [8], [19]–[22]. Class Imbalance is a key topic in recent ML research, since imbalanced class distributions negatively affect learning algorithm performances awarding the most populated classes and punishing the underrepresented ones.

In this work, we provide a thoroughly study on a wide number of solutions to Class Imbalance for data traffic extracted from different network environments and dates, which present dissimilar levels of imbalance. The most challenging traces was captured recently from an ISP backbone; meanwhile, the rest of datasets were extracted from a lab network in which users´ activities were manually simulated. Between the algorithms studied here to confront Class Imbalance, we include: six ensemble algorithms that include resampling during their training being two of them original contributions of this work; 21 well-known resampling algorithms and one well-known cost-sensitive approach. Throughout our experiments, we have applied novel methodological aspects that are gaining a special relevance due to their goodness for imbalanced problems, and they have not been employed in ML-based NTC yet, such as: the validation approach DOB-SCV or the performance metrics assumed. As an extra contribution of our research, we make publicly available our algorithm implementations in order to share them with other researchers. Some authors have already studied some solutions to Class Imbalance for NTC datasets [8], [21]–[23]**;** however, none of them employed a suitable cross-validation approach to minimize covariate shift between samples in validation folds. Furthermore, many of them employed outdated data, did not assume an early NTC approach and/or only considered TCP flows and excluded UDP traffic. To the best of our knowledge, the most of techniques considered in our experiments have not been explored for early ML-based NTC.

This article is structured as follows. Section 2 introduces Class Imbalance and reviews the most recent NTC literature. The methodological aspects applied in our experiments are presented at Section 2 along with a discussion on Class Imbalance for our datasets. During our experiments we have assessed both global and per-class performance metrics, and a novel ML validation approach (DOB-SCV) have been used to validate our results. Section 4 presents and discusses the results obtained from the experiments we have carried out. Firstly, we show and discuss the effect of the imbalanced class distributions on a base estimator, which is afterwards selected as baseline for the algorithm comparison. Secondly, we have compared a wide number of techniques for Class Imbalance evaluating their performances in terms of global metrics and statistically validating the outcomes. Thirdly, the most interesting algorithms are selected in order to thoroughly analyze their performances for each individual traffic class. Finally, Section 5 states the conclusions of this work and presents future work lines.

2. Previous work

As aforementioned, many research efforts have been focused on addressing the problem of Class Imbalance for ML problems. Through this section, we firstly provide an introductory view of Class Imbalance, and afterwards we briefly review the recent advances in ML-based NTC to state an illustrative background.

2.1 Confronting Class Imbalance

A wide number of real-world problems addressed with supervised learning fulfill the condition to be categorized as imbalanced problems, which has motivated the research on solutions to evade Class Imbalance [15]–[18], [23]. A two-class dataset is denoted as imbalanced when a class (majority class) has more instances than the other (minority class). Standard learning algorithms were designed under the assumption that labels are equally distributed in training datasets biasing the classifier performances towards the majority class. Different solutions have been proposed in order to correct the negative effects of Class Imbalance, a thorough study on many of them is provided in [24]. V. López et al. examined Class Imbalance focusing on useful performance metrics and the reasons that lead to performance losses in imbalanced scenarios (overlapping regions, small disjuncts, noisy data, …). Additionally, the authors carried out several experiments to assess the existing solutions on different binary datasets. A**s** Fig 1 shows, the existing techniques to confront Class Imbalance are categorized in three main levels according to how they address the problem:

Data Level: Data-Level methods address Class Imbalance via modifying class distributions before training, they are also known as resampling algorithms. In order to offset the class populations they create new minority samples and/or remove the existing majority ones from the original dataset. In the first case we refer to oversampling methods [25]–[27], meanwhile the techniques that reduce the number of majority samples are known as undersampling algorithms [28]–[34]. Also hybrid algorithms, which combine oversampling and undersampling, have been proposed [35], [36].

Algorithm Level: This approach includes learning algorithms that are able to award the minority class and punish the majority while training. In this instance, modified versions of learning algorithms have been proposed to tackle imbalanced distributions. Some algorithm-level approaches gaining in prominence are the ensemble techniques that incorporate a resampling phase while creating ensembles [37]–[39]**.**

Cost-sensitive Level: In this approach the algorithms learn taking into account for costs associated with the different classes [40]. Thereby, a high misclassification cost is assigned to the minority class strengthening its importance in the learning process; on the contrary, the majority class is weakened. The human perception of the problem is essential for assigning classification costs in this approach, which could lead to human errors in some cases. There mainly exist two approaches to cost-sensitive learning: (1) Direct Methods use costs directly associated with each class; meanwhile, (2) Meta-learning employs pre-processing (usually data-level techniques) and/or post-processing steps during algorithm training.

Some authors have compared some of the former solutions in their respective areas. For example, O. Loyola-González et al. [23] recently studied how resampling methods affect pattern-based classifier performances. The authors advertised about misleading results when global accuracy is employed as performance metric, and also they proved the advantages of resampling algorithms.

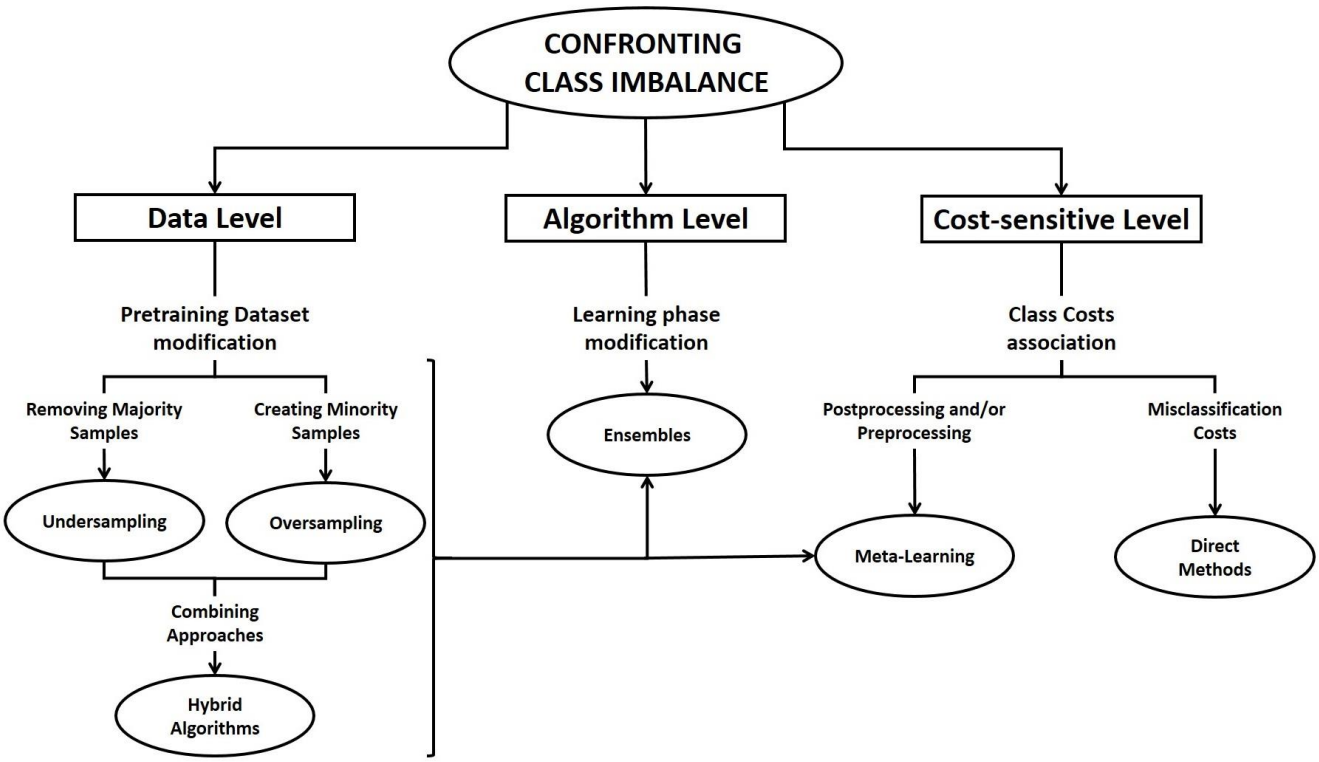

Fig. 1. Categorization of solutions to Class Imbalance

An emerging discussion in Class Imbalance is how to adapt the proposed solutions, which have been primarily designed for binary problems, to multiclass problems [30], [41], [42]. The difficulty of dealing with multiclass imbalanced problems is quite superior to learning from imbalanced binary datasets as it is shown in [43]. Decomposition techniques have attached a relevant prominence in order to adapt two-class algorithms to multiclass problems. These data preprocessing techniques transform the multiclass problem in several binary sub-problems and once the problem has been simplified, algorithms are employed in all of the sub-problems to offset Multiclass Imbalance. The most popular approaches to decompose a multiclass problem are One-versus-One (OvO) [44] and One-versus-All (OvA) [45].

Both decomposition methods have been studied by several authors. An extended analysis of imbalanced multiclass problems is provided in [41]. The authors studied the multi-minority and multi-majority effects over different performance metrics using artificial datasets and Decision Tree as base learner. Additionally, Wang et al. compared some data-level and algorithm-level techniques for 12 real-world datasets. A comparison between well-known oversampling and undersampling algorithms along with a cost-sensitive approach was carried out in [42]. The authors evaluated three state-of-art ML classifiers (Support Vector Machines, Decision Trees and K-Nearest Neighbors) in terms of average per-class accuracies and applying both OvO and OvA decomposition methods over 20 real-world problems. The obtained results reveal that oversampling techniques often provide better results than undersampling, and confirmed the advantages of applying decomposition techniques to Multiclass Imbalance. Charte et al. studied several resampling methods over different multilabel datasets in [30]. They combined simple random undersampling and oversampling along with a complex minority and majority search schemes. Furthermore, they presented measures to quantify Class Imbalance in multilabel datasets.

Another active discussion in Class Imbalance is how to validate predictive models correctly. An interesting review on performance metrics to validate classifiers in imbalanced problems is provided in [46]**.** Regarding the validation approach, some traditional methods have shown to be inefficient to validate classifiers under imbalanced conditions as it was pointed out in the work [47], in which J. G. Moreno-Torres et al. analyzed different traditional cross-validation approaches for imbalanced problems. In addition, the authors proposed a novel validation approach called DOB-SCV (Distribution Optimally Balanced Stratified Cross Validation), which is more resilient to covariate shift due to random selections. The advantages of employing DOB-SCV was afterwards confirmed in [48] through several experiments over different learning algorithms and datasets extracted from different research fields. Thus, we have assumed this validation approach for our experiments.

The particular characteristics of Internet networks lead to a high level of Class Imbalance when NTC datasets are constructed as we discuss for two different scenarios at Section 3.3.2. In this work, we study a wide number of techniques to boost algorithm performances in imbalanced NTC, including 21 data-level techniques, six ensembles techniques and one cost-sensitive approach. Amongst these algorithms, two new ensemble techniques are analyzed based on the combination of Tomek Links and ROS with boosting learning (Section 3.4). Additionally, this work constitutes a real-world case of study in which several novel methodology aspects are applied at first time in NTC. Below, we briefly review some relevant works on ML-based NTC to introduce readers to the state of the art.

2.2 Recent Advances in ML-based NTC

As aforementioned, ML has opened promising prospects in NTC and a wide number of researchers have attached their attention on this approach. One of the most important contributors to ML-based NTC was Bernialle at el. with their manuscripts [49], [50]. They presented the concept of early traffic identification, which consists in flow-based classification processing only a few number of packets at the beginning of TCP connections. The proposed classification approach accomplished satisfactory accuracies using only five packets per flow and clustering-based algorithms. Another work that discusses the effective number of packets to consider for accurate early classification is [6]. L. Peng et al. built their datasets using ordered sequence of packet sizes considering only TCP bidirectional flows. The authors reported accuracies greater than 90% using only the first 5-7 packet-sizes as predictors. W. Li and A.W. Moore [51] also experimented varying the number of packets employed to conform their datasets. They not only measured the performances of classifiers based on accuracy, but also they studied the latency in training and classification. The C4.5 Decision Tree algorithm was reported as a promising technique for NTC due to its low latency and its high accuracy.

Other authors have compared different state-of-the-art algorithms for NTC datasets. The earliest comparative study amongst ML algorithms was presented in [52]. Williams et al. confirmed the observations provided in [44], which reported Decision Trees as one of the most suitable learning algorithms for real-time NTC. Furthermore, they studied the behavior of correlation-based feature selection algorithms on their datasets showing that reducing the number of predictive attributes speeds up learning and classification without significant performances losses. Soysal and Schmidt [53] also provided a comparison between different ML algorithms confirming that Decision Trees outperform other approaches in terms of per-class precision and recall. As an additional contribution of their work, the authors studied how class distributions and errors in labeling connection flows affect classifier performances. Also, we carried out a comparison amongst ensemble algorithms using Decision Tree as base estimator in [54]. We assessed several popular ensemble algorithms showing their advantages in terms of accuracy but, also, their penalties in latency. To address the latency degradation, we presented a novel ensemble structure called T-DTC, which consists in a sequential chain of estimators acting as filters of their respective successors. T-DTC exhibited promising performances in terms of latency and accuracy over datasets extracted from two different network environments. Other authors have proposed other traffic classification approaches using different state-of-the-art learning algorithms, such as: Naïve Bayes classifier in [55]; Bayesian Neural Networks in [56]; and Support Vector Machines in [9], [57].

A current tendency in ML-based NTC is contributing to open research lines proposing ad-hoc classifiers. In the instance of [5], the authors faced the problem of detecting zero-day applications and proposed a classification approach able to detect emerging traffic and retrain itself to classify it. The proposed algorithm is composed essentially by three modules, an Unknown Discovery module, a Bag-of-Flows based classifier and a System Update module. Another classification approach with the capacity of self-learning, called Self-Learning Intelligent Classifier (SLIC), was presented in [58]. SLIC dynamically builds a training dataset and retrains a predictive model based on K-Nearest Neighbors when a new sample is introduced in the dataset. The results reported show how classification accuracy increases in each retraining iteration. The issue of performance deterioration over distant-based classifiers due to Internet dynamic conditions is analyzed in [59]. J. Camacho et al. assessed the generalization ability of 1-Nearest Neighbor in dynamic contexts, and proposed a flow pairing technique for traffic classification based on a similarity function to address this issue. Furthermore, the authors extended their experiments for P2P traffic identification.

Concerning Class Imbalance, some authors have tried to provide solutions for imbalanced NTC datasets. A class-oriented feature selection (COFS) and an ensemble learning approach are proposed in [7] to cope with non-uniform traffic distributions. COFS combines local and global metrics to remove redundant and irrelevant features outperforming traditional feature selection techniques. The presented ensemble scheme is composed by several base learners per traffic class and a subsequent weighted voting. Two simple data-level algorithms and one cost-sensitive approach (MetaCost) were compared in [22] for datasets extracted from network traces captured between 2003-2007. The authors applied Random Undersampling and Oversampling using a new strategy in order to detect minority and majority classes and set the ratios between classes. In the instance of MetaCost, the cost coefficients were adjusted according to a strategy based on flow-ratio. The reported results show how resampling algorithms can be very effective when there are insufficient training samples and cost-sensitive when there are enough number of samples. Finally, undersampling provided other interesting advantages, such as fast execution and training times. Wei H. et al. [21] also tackled the problem of class imbalanced for real-time NTC comparing several ensemble techniques that combine data sampling algorithms with boosting. The authors also proposed a hybrid approach called BalancedBoost, which is quite similar to other ensemble algorithms considered in this work. BalancedBoost outperformed the rest of algorithms using the UNIBS datasets, which is composed by traffic generated only by target hosts. Recently a cost-sensitive algorithm based on data gravitation-based classifier (IDGC) has been proposed in [8] to mitigate Class Imbalance in NTC. IDGC is a modification of the algorithm DGC proposed in [60], which introduce sensitiveness to imbalanced class distributions via applying a weighting phase using ratios between classes. Peng et al. showed that IDGC overcomes other ensemble and cost-sensitive methods focusing only on TCP connections and transforming multiclass NTC in simpler two-class datasets. Finally, we suggest reading the surveys [1]–[4] to get a more general view of NTC.

A large proportion of the above articles reported about imbalanced distributions in NTC datasets, however the works that tackle this issue are scarce. Throughout this article, we discuss Class Imbalance over real-world NTC datasets in order to insightfully analyze this problem. Additionally, the absence of studies conducting experiments to assess the benefits of solutions to Class Imbalance in early NTC encourages us to provide a uniform comparison among a wide number of these algorithms. The experiments presented below were conducted employing the most sophisticated validation approach and performance metrics for imbalanced problems up to date. The experiments were conducted employing different datasets composed by TCP and UDP traffic and extracted from two different environments, which present dissimilar Class Imbalance conditions. The classification task is faced a multiclass perspective, so that we had to adapt techniques preliminary designed for two-class problems to multiclass datasets. As part of the contributions of this work, we make our implementations available for the research community.

3. Material and Methods

The methodology followed in our experiments is described in detail through this section. Figure 2 depicts the methodology overview applied to all our NTC datasets. During dataset creation, the network traces were processed to generate a collection of 77 statistical attributes over each Internet connection assuming a flow-based classification approach. A detailed description of this process is provided at Section 3.3 along with a discussion on Class Imbalance in our datasets. After creating the NTC datasets, we applied the DOB-SCV approach to generate folds of instances that were used to train and validate the traffic classifiers, and the same folds were employed for all algorithms studied. As it was discussed in [47] and [48], traditional validation approaches, which rely on naïve random selection of samples, normally present a high covariate shift in the generated validation folds. Instead of a random selection, DOB-SCV exploits more information keeping the data distributions quite similar between folds, and thus minimizing covariate shift among folds. We generated five folds so that one fold was used to train the algorithms and the rest to validate the predictive model generated during each validation epoch. All results reported in Section 4 are the average scores obtained over the five validation folds.

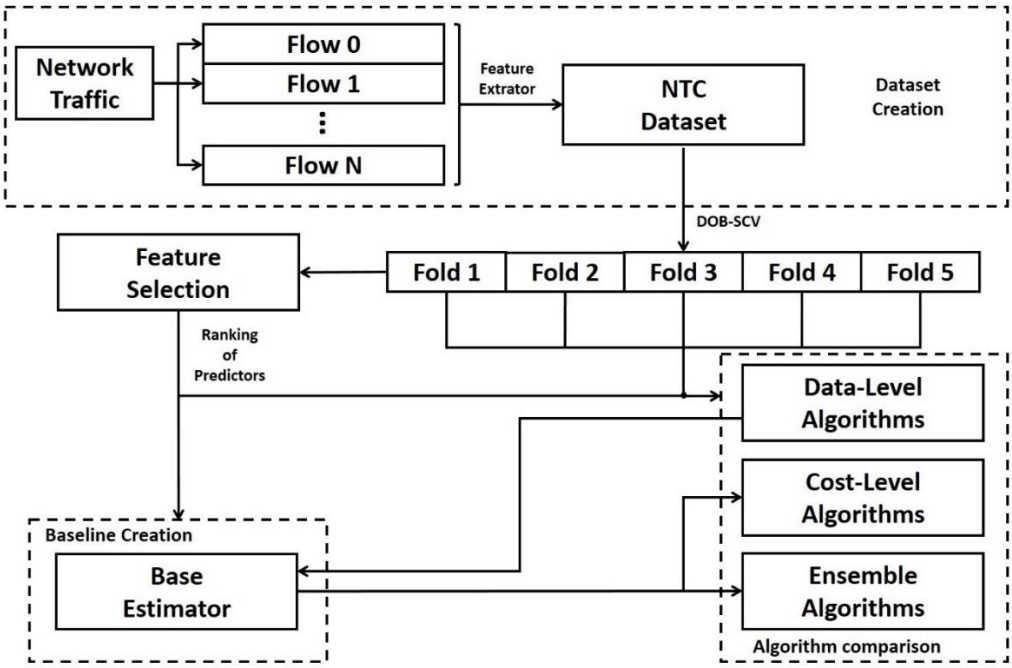

Fig. 2 Methodology Overview

Only Fold 1 was supplied to a Feature Selection (FS) algorithm in order to rank the most relevant predictors for our problem. The FS algorithm employed, called FCBFiP, is a modified version of the popular Fast Correlation Based Feature Selection algorithm, which speeds up the selection process via modifying the search strategy. We presented this algorithm and validated it against several datasets in [61]. Additionally, this algorithm was previously used in our work [54] and it is publicly available in [62]. Through FS, we generated a ranking of predictors that was applied to each fold so as to reduce the attribute space. For our experiments, we considered subset sizes from 2 to 20 with steps of 2 features in order to assess the solutions to Class Imbalance against different subset sizes.

Our main contributions are achieved essentially through two experiments. Firstly, we employed a base estimator (described at Section 3.1) to generate a baseline and compare all techniques to it. The same base estimator was afterwards employed during the comparison of solutions to Class Imbalance as Figure 2 illustrates. In the case of data-level algorithms, each fold was resampled before being used to train the base estimator, meanwhile the rest of folds were kept unaltered for validation. For ensemble and Cost-Sensitive algorithms, the base estimator was the core of the learning process. After obtaining the results, we analyzed algorithm performances according to several global performance metrics and statistically validated the outcomes to extract general observations over all datasets (Section 4.2). Finally, we observe per-class metrics for the most promising techniques on the most challenging dataset at Section 4.3 so as to confirm that the studied solutions reinforce the predictiveness on minority classes.

The algorithms to deal with Class Imbalance were collected from different sources. The data-level and two of the ensemble techniques studied are available in the Python Library imbalance-learn [63]. The boosting ensemble approaches employed are adapted versions to multiclass problems of some algorithms provided by a third party. In order to make these algorithms suitable for multiclass problems, we have designed different strategies to assist the learning process in managing ratios between classes. In total we have compared 21 Data-Level algorithms, six ensemble algorithms and one Cost-Level approach; we make accessible our implementations to the research community in [64], which constitutes an additional contribution of this work. A more detailed description of all techniques and the strategies assumed to adjust class ratios, associate classification costs with classes and assist the ensemble learning process is provided at Section 3.4. The algorithm comparison was performed in terms of several global and per-class performance metrics, which are introduced and described in Section 3.2.

3.1 Estimator choice: CART Decision Tree

During the first years of research on ML-based NTC many researchers focused on learning algorithm comparisons to find out which are the most effective learning approaches. Decision Tree has shown as one of the most suitable algorithms for online NTC due to the fact that it retains an excellent ratio between classification performances and latency [2], [51], [52]. In these works the authors shown how Decision Trees outperformed other learning approaches, such as SVM, Neural Networks and Naïve Bayes.

CART Decision tree is a learning algorithm that iteratively creates decision rules by splitting the attributes space according to an information-based criterion, normally trying to minimize metrics such as Information Gain or GINI Impurity. When Decision Trees are trained, their internal structures implement a hierarchical set of rules that looks like a tree, as Figure 3 shows for two different cases. Each level in the tree is a conditional split that describes decision regions to classify unknown samples. New unknown samples go through this hierarchical set of heuristics until they reach the final leaf, in which they are finally classified. The final class is assigned according to the classes that mostly populates the decision region. Figure 2 depicts the structure of two trained CART Decision Trees in two different conditions of Class Imbalance. In Figure 2-a, the training dataset kept an almost uniform class distribution, on the contrary, the tree (b) was trained under high Class Imbalance. Observing the bottom levels of the tree (a), we find that 127 C1 samples were correctly modeled of a total of 170, 104 C2 samples of 167, and 142 C3 samples of a total of 163. In the instance of tree (b), none of the C1 samples were correctly modeled, and only three C2 samples of a total of 26 did, whereas 453 C3 samples from a total of 462 were accurately modeled.

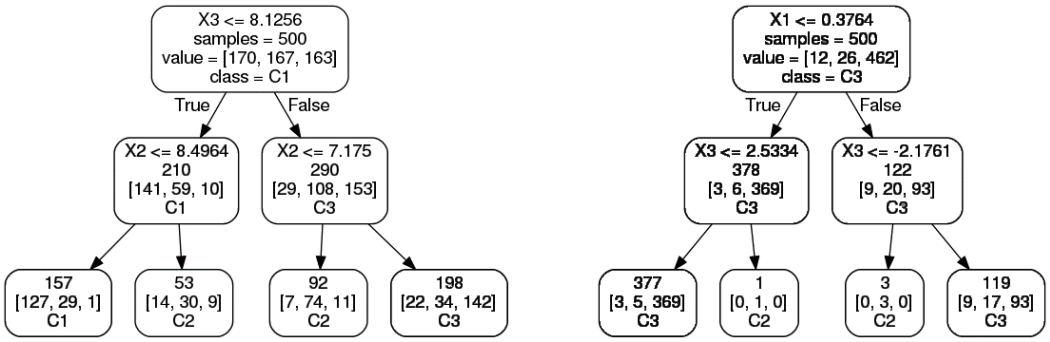

Fig. 3. Internal set of decision rules implemented by Decision Tree. The classes to predict are C1, C2 and C3; and the predictors are X1, X2 and X3

In spite of Class Imbalance sensitivity, Decision Tree algorithms have been widely employed in NTC research, and consequently we have chosen the CART Decision Tree algorithm implemented in [65] as base estimator. The CART decision Tree we have employed in our experiments tries to minimize the Gini Impurity. Gini Impurity is defined by Equation 1, where p_i is the probability for each class and C is the number of classes.

$$I_G = 1 - \sum_{i=1}^{C} {p_i}^2 \qquad (1)$$

This measure is quite sensitive to Class distributions, since I_G is computed using the square root of class probabilities found in the training dataset. Therefore, if the initial dataset is highly imbalanced, this metric will bias towards the most populated classes. The Class Imbalance sensitivity of CART Decision Tree makes it a good base estimator to assess the enhancements provided by the techniques studied.

3.2 Performance Metrics:

Which performance metrics use when an imbalanced problem is faced is already an open research topic in ML. Traditional metrics that measure the overall classifier performances were designed without considering Class Imbalance. Thus, no every assessment metric is appropriate for validating learning systems in this context [46]. In order to consistently compare the performances of the different solutions to Class Imbalance, both global and per-class metrics are assumed. We consider global metrics quite worthy to figure out the performances of classifiers on the whole network traffic. Additionally, per-class metrics describe the behavior of the algorithms on individual classes so that they are very insightful to know if minority classes are really strengthened. Below, the per-class and global metrics used for our comparison are introduced. Finally, we introduce other measures to assess the level of Class Imbalance in our datasets, and the statistical approach used so as to validate the results obtained in the comparison.

3.2.1 Per-class Metrics: Class Accuracies and AUC-ROC

The techniques to mitigate Class Imbalance are expected to reinforce the predictive power on minority classes and, eventually, weaken the majority classes. Therefore, it is crucial to evaluate the classifiers in terms of metrics that describe the performances on individual classes. To this aim, we assume per-class accuracies and AUC-ROCs (Area Under Curve – Receiving Operating Characteristics). The former is a general metric and it is defined by Equation 2, where TP_i denotes the true positives on samples belonging to class i (note that ACC_i is similar to per-class recall [46]). The latter is a scalar metric computed from the ROC curve. ROC curve is a graphical representation of binary classifier performances in terms of true positives and false positives. We have extended this binary metric to multiclass problems using One-versus-All approach. AUC-ROC method is quite interesting for imbalanced datasets, since it measures the quality of classifiers irrespective of class distributions.

$$ACC_i = \frac{TP_i}{\#Samples\ of\ Class\ i} \quad (2)$$

In order not to collapse the result section due to the high number of algorithms considered, we only present and discuss the per-class metrics for the base estimator (Section 4.1) and the most interesting algorithms (Section 4.3).

3.2.2 Overall Metrics: Overall, Byte, Average Accuracies & Multiclass AUC-ROC

Global performances for classifiers are often assessed by Overall Accuracy (OA), OA measures the percentage of samples correctly labeled as Equation 3 describes. TP_i denotes the number of true positives on class i and $\#Samples$ the total number of instances contained in the dataset. Since flow-level classification is assumed, OA can be considered as flow accuracy.

$$OA = \frac{\sum TP_i}{\#Samples} \quad (3)$$

Other interesting performance is the Byte Accuracy (BA) defined by Equation 4. Each Internet connection consumes network resources in terms of duration, bytes and number of packets transferred. From a network management perspective, measuring the quantity of bytes correctly classified is quite reveling to figure out the quality of traffic classifiers. Thus, we report the BA score in the result section, which is the percentage of bytes accurately classified over the total number of bytes contained in network traces.

$$BA = \frac{Bytes\ classified\ correctly}{Total\ bytes\ captured} \quad (4)$$

Both OA and BA metrics are quite sensitive to Class Imbalance. If a class accumulates the most of instances and/or the most of bytes transferred, OA and BA are not representative metrics for the rest of minority classes. Satisfactory accuracies on majority classes could mask poor classification rates on the minorities. To avoid misleading observations, we have evaluated two additional well-known metrics that accurately describe the quality of classifiers for imbalanced problems. A reveling metric for imbalanced problems is G-mean (GM), which is the geometric mean of all per-class accuracies (or recalls [46]). GM for a problem comprising n classes is defined in Equation 5. One strategy to extend per-class metrics to multiclass metrics that summarize them is the macro averaging. The Macro-Average is the arithmetic mean of metrics partially computed for each individual class. This metric has shown more proper for imbalanced datasets than other global scores, since the impacts of minority and majority classes over the final score are the same. Therefore, we assume the Multiclass AUC (MAUC), which is defined by Equation 6 for n classes.

$$GM = \sqrt[n]{\prod ACC_i} \quad (5)$$

$$MAUC = \frac{\sum AUC_i}{n} \quad (6)$$

3.2.3 Measuring the imbalance level: imbalance ratio per label

An assessment approach to measure the level of Class Imbalance in multilabel datasets was presented in [30]. This approach is based on the imbalance ratio per label (IRLbl) defined by Equation 7, which is the ratio between the number of majority samples and the number of samples belonging to a given class i. Thereby, IRLbl for the majority class will be 1, meanwhile it will be larger for minority classes.

$$IRLbl(i) = \frac{\#Samples\ of\ majority\ class}{\#Samples\ of\ class\ i} \quad (7)$$

Once the IRLbl has been computed for each class, the mean and variance of all IRLbl values are computed to get general information about Class Imbalance in the whole dataset. The larger the mean of IRLbl, the higher the level of imbalance in the dataset; and the larger the variance, the higher the difference among class populations. We assume these metrics so as to figure out the level of difficulty imposed by imbalanced class distributions in our datasets.

3.2.4 Statistical Validation

In our second experiment we compare a wide number of resampling algorithms according to several global metrics over four datasets. When algorithms are compared using different datasets, the statistical significance must be verified to assure that the obtained results are consistent [66]. A well-known method to compare a set of algorithms against different datasets is Friedman's Test. Friedman's Test is a non-parametric statistical method, which sets as null hypothesis that all algorithms involved in the comparison achieve the same performances: in short, no statistical differences exist between them. In order to confirm or reject the null hypothesis, algorithms are ranked for each dataset according to their performances, and the position that each algorithm occupies in the ranking is assigned as scores. Then, Friedman's score is computed as Equation 8 describes, being k the number of algorithms in comparison, N the number of datasets and R_j the score obtained by each algorithm for the dataset j.

$$\chi^2{}_F = \frac{12N}{k*(k+1)}\left[\sum_j R_j^2 - 0.25k*(k+1)^2\right] \quad (8)$$

Once χ^2_F is computed, the associated p-value is obtained from a chi-squared random distribution with $k-1$ degrees of freedom. The lesser the resulting p-value, the greater the probability that statistical significance exists between the algorithms.

3.3 Datasets: Network Environments, Feature Extraction & Level of Class Imbalance

Internet networks environments normally differ each other in many features, such as: the kind of traffic observed, the quantity of connections belonging to each application, the topologies and traffic rates. These facts considerably affect the predictors contained in NTC datasets. Traffic rates could affect predictors related to Inter-Arrival Times, and network topologies may carry packet losses or multipath effect that influence the values of NTC predictors. Consequently, it is highly recommended to validate ML-based traffic classifiers in several network scenarios. We have selected four network traffic captures collected from two different network environments: a lab network and ISP backbone network. Table 1 includes relevant information about the network traces employed in our experiments.

Privacy policies normally hinder the possibility of getting third-party real network traces. To evade this constraint, the CBA research group of UPC BarcelonaTech generated network traffic for research purposes in their lab. They manually simulated host activities for a long term and captured the network traffic generated in the hosts to assess DPI tools [67]. The datasets resulted from processing these network captures have been called HOST datasets in this work.

In addition to HOST data, we have included datasets collected from a much more challenging scenario. An Internet Service Provider, which provide Internet to more than two million of users across Spain, has cooperated in this research sharing real network traffic with research purposes. The network traffic was captured recently in a node of their backbone network where traffic rates of 7 GB/s are supported. These datasets have been called ISP traces in our result section. The name of the ISP is omitted in this work due to security concerns.

3.3.1 Feature Extraction: Statistical Attributes & Labeling

The datasets involved in our experiments include 77 statistical attributes processing only five packets at the beginning of each Internet connection. Computing the attributes using a limited number of packets assures that our classifiers fulfil the early classification requirement presented in [49]. The classification objects considered are bidirectional flows, therefore each flow sample contains information about ingoing and outgoing packets. The complete list of predictors is available at an Annex in our previous article [54].

As we are assuming a supervised approach for our classification problem, we need to consistently associate each connection flow to the application that generates it. There are several fashions to label instances for NTC datasets, but it is highly recommended to employ a DPI approach due to their high accuracy. Since the tool nDPI [13], publicly available at [68], has shown as one of the most accurate open source DPI tool and it is able to handle encrypted traffic [69], we used it to label our datasets.

The tool nDPI classifies application flows with an excessive fine granularity, which turns out datasets with an unmanageable number of classes. Evaluating Class Imbalance solutions on a high number of classes leads to too heavy execution times and a major challenge when ratios between classes are adjusted for resampling techniques. Additionally, some learning algorithms are pretty sensitive to the number of classes, hindering classifiers performances when they deal with a vast number of classes to predict. In order to avoid the former constraints, we have assumed an application grouping strategy, in which applications and protocols that share similar features are clustered in more general descriptive objects. Application grouping was introduced in [70], and this strategy has been commonly applied in numerous relevant ML-based NTC works [7], [10], [22], [51], [55], [56], [71].

Table 1. Network Traffic Traces Information. $\overline{IRLbL}$ denotes the mean of IRLbl metric and $\sigma(IRLbL)$ denotes its variance

	Start date	Duration	Datasize	# Packets	# Flows	$\overline{IRLbL}$	$\sigma(IRLbL)$
ISP-1	17/01/2017	298 seconds	12.12 GB	8863530	231137	38.50	35.79
ISP-2	23/03/2017	600 seconds	35.62 GB	33156082	627898	91.22	107.45
HOST-1	25/02/2013	~59 days	9438 MB	5062825	121293	4.42	3.15
HOST-2	25/02/2013	~32 days	22 GB	21000000	245627	17.29	18.28

Table 2. Network Application distribution for our datasets. %I denotes the percentage of instances belonging to each class and %B denotes the percentage of bytes transferred by each application in the network captures.

	P2P		WWW		DNS		INT		S/C		BULK		Media		E/C		QUIC	
	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B
ISP-1	-	-	72.60	91.30	21.00	0.09	2.45	0.11	0.66	0.16	-	-	-	-	1.59	0.44	1.70	8.10
ISP-2	0.25	<0.01	70.20	85.70	21.90	0.21	2.57	0.41	0.90	0.24	-	-	0.26	0.10	1.59	0.25	2.33	13.40
HOST-1	33.00	15.90	32.83	27.61	9.12	0.09	10.30	2.73	5.96	0.06	5.72	23.71	3.07	29.9	-	-	-	-
HOST-2	14.30	7.90	17.10	11.80	7.21	0.04	55.40	67.1	1.06	0.01	3.43	6.22	1.50	6.93	-	-	-	-

The WWW class is composed by HTTP and HTTPS queries towards many diverse websites. The DPI tool employed to label the dataset is able to directly detect connections to the most popular web services (such Google, YouTube, Facebook and so on), however some HTTPS connections were labeled as SSL on port 443. These instances were also mapped to the WWW class. Other website queries are represented by QUIC class, QUIC is a recent transport protocol implemented by the browser Google Chrome whose presence in the ISP traces is quite relevant. The eDonkey, Torrent and other peer-to-peer traffic have been grouped into P2P class. DNS protocol has been found with a notable presence in HOST and ISP data, thereby this protocol was considered as

an independent class. Media groups applications and protocols as RTP and Skype. Remote control protocols as SSH, Telnet and others were represented by the class interactive (INT). The network service protocols (such as NetBios, Radius, Kerberos and so on) have been grouped in the class Service/Control (S/C). The Email/Chat class includes applications as WhatsApp, email services and so on. Finally, Bulk traffic groups File transfer protocols, such as FTP. NDPI reported some connection flows as unknown, so that we used the port numbers (IANA) to assign the final application class in these cases. If it was not possible to identify the application for any flow, these samples were excluded from the datasets. Other applications groups, as database queries and online games, were found in our traffic data; however, we excluded them from our experiments due to their hugely weak presence in the datasets. The datasets used in our experiments are accessible to the research community via emailing the authors. Table 2 contains the populations found in the datasets.

3.3.2 Level of Class Imbalance in our datasets

Table 2 contains the class distributions found in our datasets in terms of number of flows and the bytes consumed by each group of applications. In the instance of ISP traces, the majority classes are WWW and DNS, which accumulate more than 90% of the samples contained in both datasets. On the contrary, we found that the minorities are INT, S/C, E/C and QUIC for both, and also MEDIA and P2P in the case of ISP-2. In spite of the different capture durations and dates (Table 1), the distributions of classes are very similar to each other, but with the main difference that P2P and Media traffic emerged in ISP-2 with a quite low sample representation. This fact affects the metrics used to assess the level of Class Imbalance, note that $\overline{IRLbL}$ and $\sigma(IRLbL)$ for ISP-2 are much larger than the ISP-1 (Table 1). Focusing on the byte populations for ISP traces, we found that QUIC takes an important relevance. Although QUIC has a weak presence in terms of %I, it consumed more than the 8% of bytes for ISP-1 and more than the 13% for ISP-2. However, WWW is remaining being the most byte-consuming for both datasets. Regarding HOST datasets, we find that they present a lesser degree of imbalance than the ISP traces. This fact is caused by the differences between network environments, since ISP traffic aggregates connections flows coming from many users, meanwhile HOST traces were captured in host computers.

The level of Class Imbalance in HOST-1 is much lower than HOST-2 as can be noted observing $\overline{IRLbL}$ and $\sigma(IRLbL)$ from Table 1. In the instance of HOST-1, P2P and WWW are the majority classes summing up more than the 60% of the samples, meanwhile MEDIA is the lowest populated class with only the 3.07% of samples, followed by S/C and BULK with a percentage of samples close to 6% each one. Note that, although MEDIA and BULK flows do not have a relevant presence in HOST-1 in terms of samples, these applications accumulate near the 60% of bytes. For this network trace, P2P and WWW also consumed an important percentage of bytes, meanwhile DNS, INT and S/C consumed much less. In the case of HOST-2, INT is remarkably the most populated class having more than 55% of samples. Contrary, the most underrepresented classes in terms of instances for HOST-2 are S/C and INT with a 1.06% and 1.5% of instances respectively. The high differences between the majority and the minority classes cause that HOST-2 presents a greater level of Class Imbalance than HOST-1. In terms of percentage of bytes for HOST-2, INT is the most byte-consuming application with more than the 67% followed by WWW, P2P, MEDIA and BULK, which add more than 30% of bytes. DNS and S/C are very light in terms of bytes captured in the network trace.

As we have noted, Class Imbalance have an important presence in our datasets presenting multi-majority and multi-minority classes. Below, we introduce the algorithms studied and the multiclass strategies to confront Class Imbalance.

3.4 Algorithms & Strategies to confront Class Imbalance

In this section we introduce the algorithms employed in our experiments and the strategies assumed to tune their parameters. Table 3 contains a brief description of each algorithm and Figure 4 shows the strategies applied. As part of the contributions provided in this work, the algorithms we have implemented are accessible to the research community in [64].

We have collected several techniques from different approaches to confront Class Imbalance: 21 data-level algorithms, including undersampling, oversampling and hybrid approaches; 6 algorithm-level techniques and one well-known cost-sensitive approach. All data-level techniques along with Easy Ensemble and Balance Cascade algorithms are implemented in the Python library imbalanced-learn [63]. The other ensemble schemes are two-fold contributions from a third party and ours. The algorithms SMOTEboost and RUSboost were collected from the algorithm repository [72]. These algorithms were not adapted to multiclass problems, so that we had to upgrade the implementations to deal with multiclass problems. Furthermore, we have implemented two unexplored boosting algorithms: TLboost and ROSboost, which have already not been applied to ML to the best of our knowledge. The maximum number of estimators were set to 10 for all ensemble structures, since more estimators did not yielded better results for our datasets.

Finally, we have implemented the cost-sensitive approach MetaCOST [40]. Preliminarily, we tested the strategy presented in [22] to compute the classification costs for MetaCOST, however majority classes were strongly punished due to the huge differences between the number of samples for different classes. In order to mitigate this fact, we have applied Equation 9 to compute classification costs. Thereby, the cost associated with misclassifying a sample belonging to class i as class j is $Cost_{i,j}$, where C_i denotes the number of samples for class i.

$$Cost_{i,j} = \begin{cases} log_{10}(C_i)/log_{10}(C_j) & i \neq j \\ 0 & i = j \end{cases} \quad (9)$$

NTC is a multi-minority and multi-majority problem, thus tuning manually the ratio of each class for resampling methods is a quite arduous and time-consuming task. Additionally, the boosting algorithms need a procedure to set the resampling ratios

between classes for each learning iteration. Consequently, we have designed different strategies to set the former parameters during our experiments (Figure 4). In the case of Data-Level Undersampling, majority classes are considered classes whose number of samples are greater than the mean of all populations (N_{mean}), and majority classes are undersampled until reaching N_{mean} so as to avoid excessive information removal. Regarding Data-Level Oversampling, minority classes are considered all classes with a lesser population than the majority class (N_{maj}), so that all minority classes are oversampled until equaling the majority class. In the instance of hybrid approaches, the classes with a number of samples lesser than N_{mean} were oversampled and the classes with greater populations were undersampled until reaching N_{mean}.

In the instance of ensemble algorithms, EE and BC are ensemble algorithms based on creating bags of estimators trained using balanced datasets. These algorithms state that the minorities classes resampled until equaling the most majority class. However, boosting algorithms need to implement a resampling strategy to adjust the number of classes employed in each boosting iteration. In the case of algorithms that combine boosting and undersampling (UnderBoosting), all classes with more than N_{mean} are undersampling until N_{mean}. Meanwhile, in the case of OverBoosting algorithms, majority classes are considered the classes whose number of samples are lesser than N_{maj}, and they are resampled until reaching N_{maj}. For both, Under and OverBoosting, the minority and majority classes are proportionally resampled until accomplishing the corresponding sample populations.

Table 3. Algorithm selected to deal with Class Imbalance in our NTC datasets. The strategies presented in Figure 4 were applied to the algorithms marked with an asterisk

	Algorithm Description
OVERSAMPLING	
Random OverSampling (ROS*)	The minority class is resampled by replicating samples randomly selected. This algorithm is the simplest oversampling technique.
Synthetic Minority Oversampling TEchnique (SMOTE*)	Synthetic data are generated for the minority class [25]. *K* minority nearest neighbors are selected for each minority sample, one of these neighbors is randomly chosen and one new sample is generated at a random point in the segment that joins the neighbors. This process is repeated until accomplish the desired number of new minority samples.
SMOTE with Borderline 1 and 2 (SMOTE-B1* & B2*)	This modification of SMOTE assumes that only minority samples placed near the borderline between classes are important for learning [26]. This SMOTE version detects borderline examples and strengthens them according to two strategies. In borderline 1 only *k* nearest neighbors belonging to minority class are oversampled, meanwhile both majority and minority, borderline samples are generated in SMOTE-B2.
ADAptive SYNthetic algorithm (ADASYN*)	ADASYN adaptively resamples the minority class according to the level of difficulty in the learning process [27], so as that more synthetic samples are generated for classes difficult to predict. In the generation process the algorithm randomly selects the *k* nearest neighbors around minority samples and estimate the distribution of the data. Finally, new samples are generated in middle points between minority samples and one of their neighbors randomly chosen.
UNDERSAMPLING	
Random UnderSampling (RUS*)	RUS randomly selects samples belonging to the majority classes and removes them from original datasets. RUS is the simplest approach to apply undersampling to imbalanced datasets.
Near Miss (NM-1*, 2* & 3)	Near-miss samples are defined as the majority samples that are located in minority class nearby. NM-1, 2 & 3 remove the near-miss samples according to a KNN strategy. Three strategies were developed to determine if a given sample is near-miss, all of them are described in [29].
Condensed Nearest Neighbor (CNN)	CNN iteratively finds a consistent subset with the minimal number of initial samples. CNN employs the Nearest Neighbor rule to determine if a sample will be retained or discarded.
Tomek Links (TL)	A Tomek Link consists of a pair of samples that are nearest neighbors but each one belongs to a different class [34]. TL detects and removes Tomek Links from the initial dataset.
One Sided Selection (OSS)	OSS intelligently removes the majority samples in two phases: (1) a 1-KNN classifier selects a representative subset of majority samples, and (2) the majority samples that participate in Tomek Links are removed.
Edited Nearest Neighbor (ENN)	ENN removes samples that are misclassified by a k-NN classifier [31]. The purpose of this technique is to remove outliers and overlapped samples between different classes.
Neighborhood Cleaning Rule (NCR)	NCR [32] removes noisy examples in two steps essentially: (1) NCR employs the ENN rule to identify noisy samples, and (2) noisy samples with 3 of their 5 nearest neighbors belonging to different classes are removed.
Instance Hardness Threshold (IHT)	IHT is a recent data reduction technique that trains a base classifier, estimates sample probabilities and removes the training samples whit weak probabilities [33]. We employed decision tree as base estimator for our experiments.
HYBRID SAMPLING	
SMOTE+Undersampling (SMOTE-TL*, SMOTE-ENN*)	SMOTE-TL [35] firstly oversamples minority samples using SMOTE and, afterwards, removes the TL links. Meanwhile, SMOTE-ENN [36] cleans the oversampled dataset applying ENN rule.
ENSEMBLE ALGORITHMS	
EasyEsemble (EE)	EE creates a bag of balanced datasets using ROS to train a set of base estimators, whose predictions are aggregated according majority voting [37].
BalanceCascade (BC)	BC is a supervised version of EE. BC creates a bag of balanced datasets, which are refined using a base estimator. [37]
OverBoosting (ROSboost*, SMOTEboost*)	OverBoosting oversamples minority classes in each boosting iteration. ROSboost employs ROS during learning, meanwhile SMOTEBoost oversamples the dataset using SMOTE [39].
UnderBoosting (RUSboost*, TLboost)	UnderBoosting undersamples majority classes in each boosting iteration. RUSboost [38] employs RUS during learning, meanwhile TLboost removes Tomek links in each iteration.
COST-SENSITIVE	
MetaCOST (MetaCOST)	MetaCOST is a well-established cost-sensitive technique independent from the learning algorithm employed [40]. MetaCOST creates a set of estimator trained using resampled datasets, which estimates the post-probabilities of training samples and applies classification costs to relabel the initial training set.

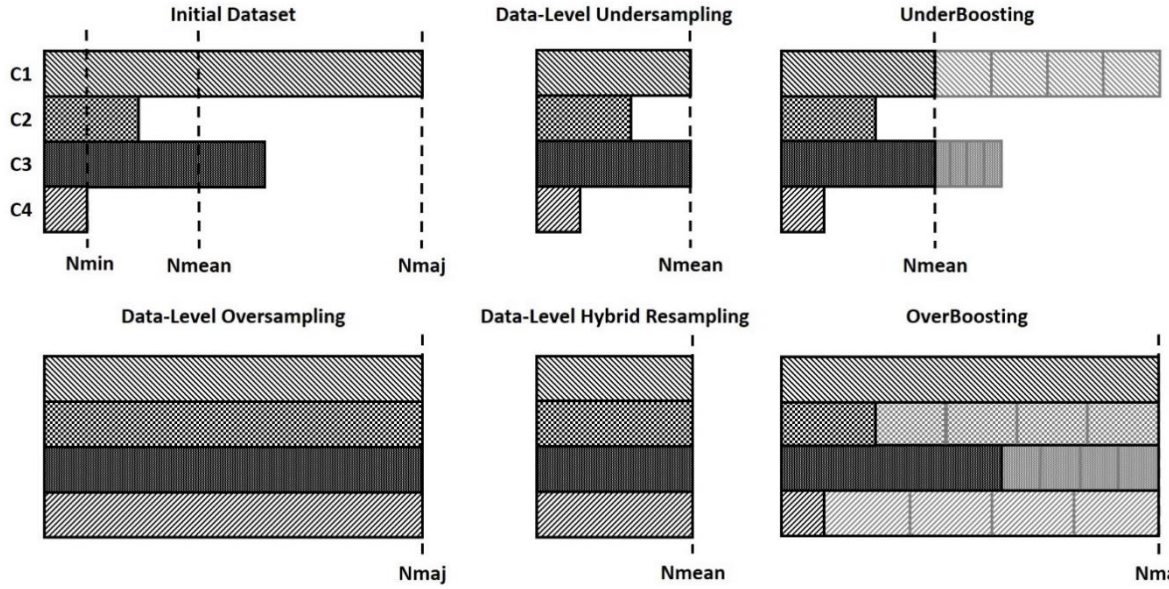

Fig. 4. Strategies to adjust resampling ratios. C1, C2, C3 and C4 denote arbitrary classes, N_{min} the minimum population, N_{maj} the maximum population and N_{mean} the mean of all populations

4. Experimental Results

Through this section we present and discuss the results obtained during our experiments. Firstly, we analyze the effect of Class Imbalance on the global and per-class metrics for our datasets using the base estimator and with the aim of establishing the baselines to compare the algorithms under study. Secondly, we compare the techniques introduced in Section 3.4 in terms of the global metrics in order to figure out which algorithms are the most proper for imbalanced NTC. Additionally, a statistical procedure is employed to extract general observations on algorithm performances over all our NTC datasets. Finally, we validate the most promising techniques in terms of per-class metrics for the most challenging dataset so as to assure that minority classes are really strengthened.

4.1 Preliminary results: Assessing Class Imbalance & Baseline

In this experiment, a CART Decision Tree was trained using the datasets presented in Section 3.3 and varying the subset sizes after reducing the attribute space. Through this evaluation, we assess the negative effect of Class Imbalance on the global and per-class metrics and establish the baselines for the subsequent algorithm comparison. Table 4 presents the global metrics resulting from this preliminary experiment, and Table 5 contains the per-class metrics.

From Table 4, it is apparent that notable differences exist between the global metrics obtained for different network scenarios. Generally, the predictive models produced for ISP datasets achieved lower performances than HOSTs. For example, the best OA for ISP-1 reached 92%, meanwhile the highest OA for HOST-1 overcame 98%. Note from Table 5 that per-class metrics for HOST datasets are also greater than for ISP datasets. These clear differences in performances suggest that ISP network environment comprises a more challenging traffic classification task than HOST. As aforementioned in Section 3.3, the ISP traces were captured in the middle of a high-speed backbone, where traffic is much more susceptible to packet losses and packets out of order.

Focusing on ISP traces, we find that the differences between ISP-1 and ISP-2 are not as large as the observed between network environments. However, the observations change depending on the performance metric we focus on. In the instance of GM, the predictive model trained with ISP-2 generally overcame ISP-1, on contrast to OA, BA and MAUC, which were slightly greater for ISP-1 than for ISP-2. Note also that there are points in which all global metrics notably increased for both datasets when the subset sizes vary, and that the performances smoothly fluctuated without high variations after those points. The abrupt performance increases happened when 6 and 8 predictors were selected for ISP-1 and ISP-2 respectively. These sharp raises are strongly related to the high improvements on WWW and DNS traffic detection, but also on other applications with lesser impacts on the class distributions, such as S/C for ISP-1 or Media and E/C for ISP-2. Another remarkable observation is that the OA and BA losses are more significant for ISP-2 than for ISP-1 when insufficient attributes were selected. This fact is directly connected to important differences in WWW per-class metrics (Table 5) amongst ISP traces, which reveals the high impact of this traffic class over OA and BA. The best models in terms of GM and MAUC were achieved using 8 and 14 predictors for ISP-1 and ISP-2 respectively. Furthermore, the best OA and BA were achieved using 16 and 18 features for ISP-1, meanwhile the subset with 18 attributes produced the best models in terms of OA and BA for ISP-2.

Regarding HOST datasets, we find that all global metrics (Table 4) fast boosted when 4 and 2 predictors were selected for HOST-1 and HOST-2 respectively. After that point, the global metrics linearly grew up until reaching a point in which they fluctuated with smooth variations when subset sizes change. In the case of HOST-1, we find from Table 5 that P2P, WWW, S/C and BULK samples were poorly detected when two predictors were selected for training. Note also that the same happened for HOST-2, but with weaker per-class metric deteriorations. In the instance of HOST-1, the best models in terms of OA and BA were produced with 18 and 14 attributes, whereas the maximum MAUC and GM were accomplished selecting 18 and 20 predictors.

While on HOST-2, the maximum OA resulted from selecting 12 or 14 features and the best BA from selecting 10. Respecting MAUC and GM for HOST-2, the former reached its maximum at 16 and the latter at 12, 14 or 18 features.

Table 4. Global metrics obtained varying the subset sizes and employing the base estimator as learner. The results are expressed in %

	ISP-1				ISP-2				HOST-1				HOST-2			
#Fe	**OA**	**BA**	**MAUC**	**GM**	**OA**	**BA**	**MAUC**	**GM**	**OA**	**BA**	**MAUC**	**GM**	**OA**	**BA**	**MAUC**	**GM**
2	74.13	75.11	77.56	42.01	44.27	45.90	78.63	58.40	77.72	81.95	87.47	77.18	95.86	94.48	94.79	89.97
4	76.97	78.67	78.49	43.22	53.54	55.80	80.21	70.15	94.39	98.25	95.47	91.53	98.20	98.41	97.47	95.14
6	90.45	91.47	91.60	84.97	63.97	66.76	83.63	74.82	95.30	96.99	96.21	92.86	99.00	99.18	98.45	97.02
8	90.94	92.11	91.95	85.57	87.86	88.35	90.46	86.53	95.43	98.27	96.26	92.94	99.01	99.17	98.42	96.96
10	91.93	93.01	91.87	85.21	87.74	88.55	90.75	87.22	96.37	98.23	96.81	94.07	99.16	99.21	98.61	97.38
12	91.73	92.44	91.85	85.00	87.90	88.84	90.87	87.54	98.21	99.24	98.60	97.43	99.23	99.00	98.80	97.68
14	91.63	92.62	91.86	85.18	88.09	88.69	91.35	88.38	98.20	99.51	98.61	97.43	99.23	99.20	98.80	97.64
16	92.42	93.19	91.83	84.91	88.23	88.73	91.23	88.07	98.39	99.43	98.69	97.62	99.20	99.06	98.78	97.69
18	92.47	93.10	91.86	85.12	88.59	89.10	91.27	87.95	98.48	99.19	98.73	97.65	99.22	98.93	98.80	97.66
20	92.36	92.85	91.84	85.28	88.54	89.10	91.30	88.13	98.46	99.46	98.72	97.68	99.22	99.19	98.77	97.63

Table 5. Per-class metrics obtained varying the subset sizes and employing the base estimator as learner. The results are expressed in %

	P2P	WWW	DNS	INT	S/C	BULK	MEDIA	E/C	QUIC
	ACC/AUC	**ACC/AUC**	**ACC/AUC**	**ACC/AUC**	**ACC/AUC**	**ACC/AUC**	**ACC/AUC**	**ACC/AUC**	**ACC/AUC**
ISP-1									
2	-/-	87.23/91.83	29.59/64.56	74.96/85.89	4.35/52.13	-/-	-/-	69.20/81.79	94.35/89.14
4	-/-	90.91/93.65	29.96/64.74	78.24/87.85	4.64/52.30	-/-	-/-	69.87/83.09	94.35/89.29
6	-/-	91.94/94.50	87.60/93.50	83.02/90.75	84.70/91.31	-/-	-/-	73.88/85.09	89.94/94.42
8	-/-	91.90/94.45	90.04/94.73	83.50/90.99	86.32/92.18	-/-	-/-	73.24/84.76	89.90/94.60
10	-/-	93.48/95.14	90.27/94.83	82.93/90.66	87.30/92.50	-/-	-/-	69.65/83.79	89.94/94.63
12	-/-	92.81/94.82	90.12/94.76	83.86/91.11	86.78/92.23	-/-	-/-	68.64/83.04	90.24/94.77
14	-/-	93.05/94.94	90.06/94.73	83.09/90.73	86.96/92.31	-/-	-/-	70.11/83.85	89.99/94.65
16	-/-	93.24/94.86	92.51/95.98	82.41/90.44	85.86/91.91	-/-	-/-	67.34/82.47	91.22/95.41
18	-/-	93.36/94.97	92.50/95.95	82.22/90.33	86.14/92.06	-/-	-/-	68.40/83.05	90.90/95.25
20	-/-	93.25/94.96	92.85/96.12	82.64/90.59	85.91/91.98	-/-	-/-	68.62/83.09	91.18/95.39
ISP-2									
2	90.45/94.24	33.02/65.48	69.02/82.68	95.48/77.57	85.10/91.49	-/-	26.63/62.38	38.18/67.77	79.44/87.46
4	89.36/93.85	43.90/70.42	77.58/87.03	85.08/78.95	85.01/91.71	-/-	35.14/63.41	36.76/67.28	80.82/89.07
6	89.68/94.10	57.61/77.69	78.94/87.87	93.84/84.76	85.55/92.17	-/-	38.60/68.11	47.87/72.77	84.99/91.55
8	90.19/94.56	89.81/93.45	83.16/91.24	86.95/91.99	88.31/93.68	-/-	64.98/81.64	70.08/83.48	88.43/93.67
10	90.64/94.77	89.44/93.37	83.58/91.45	86.59/91.80	88.12/93.60	-/-	66.32/82.24	73.13/84.97	88.57/93.76
12	90.32/94.63	89.60/93.42	83.80/91.57	86.10/91.58	88.85/93.96	-/-	67.36/82.74	73.58/85.24	88.61/93.79
14	90.45/94.70	89.68/93.58	83.92/91.64	87.65/92.22	88.92/94.01	-/-	68.75/83.46	77.17/87.19	89.03/94.00
16	90.77/94.86	89.89/93.65	84.04/91.69	87.43/92.20	88.64/93.88	-/-	68.15/83.11	76.17/86.72	88.54/93.76
18	90.58/94.76	90.28/93.85	84.39/91.85	87.76/92.49	89.10/94.12	-/-	67.23/82.71	75.72/86.50	88.76/93.87
20	90.32/94.61	90.38/93.90	83.78/91.55	87.90/92.59	89.03/94.06	-/-	68.21/83.16	75.85/86.59	88.94/93.96
HOST-1									
2	62.19/79.67	82.61/84.30	90.67/95.06	94.45/94.65	98.79/99.40	70.07/82.71	53.58/76.49	-/-	-/-
4	97.24/97.30	93.38/96.36	90.68/95.07	99.30/99.55	98.89/99.44	89.93/94.49	74.01/86.05	-/-	-/-
6	96.19/97.65	93.31/96.44	98.33/98.43	99.34/99.58	98.85/99.42	90.13/94.78	76.25/87.17	-/-	-/-
8	96.40/97.74	93.52/96.55	98.34/98.43	99.37/99.65	98.86/99.43	90.10/94.78	76.33/87.23	-/-	-/-
10	88.42/93.99	96.53/98.04	98.32/98.42	99.37/99.67	98.89/99.44	91.99/95.66	85.91/91.17	-/-	-/-
12	99.19/99.48	96.73/98.13	99.59/99.77	99.42/99.69	99.07/99.53	97.41/98.46	90.93/95.12	-/-	-/-
14	99.01/99.39	96.70/98.11	99.55/99.75	99.39/99.68	99.07/99.53	97.49/98.49	91.12/95.18	-/-	-/-
16	99.24/99.50	97.33/98.45	99.60/99.78	99.41/99.69	99.07/99.53	97.43/98.51	91.52/95.46	-/-	-/-
18	99.09/99.44	97.44/98.49	99.61/99.78	99.43/99.70	99.07/99.53	97.49/98.54	91.66/95.51	-/-	-/-
20	99.33/99.55	97.34/98.45	99.51/99.73	99.44/99.71	99.07/99.53	97.37/98.48	91.96/95.69	-/-	-/-
HOST-2									
2	88.76/94.01	92.48/95.40	94.96/97.26	99.78/99.39	95.81/97.82	85.45/92.50	74.93/87.16	-/-	-/-
4	98.26/99.05	96.26/97.86	96.94/98.30	99.67/99.81	96.62/98.25	89.14/94.38	89.65/94.63	-/-	-/-
6	99.02/99.47	96.83/98.24	99.11/99.53	99.83/99.89	96.96/98.46	95.94/97.90	91.68/95.66	-/-	-/-
8	99.07/99.50	96.92/98.28	99.09/99.51	99.82/99.88	96.96/98.46	95.95/97.90	91.19/95.42	-/-	-/-
10	99.21/99.56	97.40/98.59	99.11/99.52	99.90/99.92	96.96/98.46	96.67/98.27	92.60/96.15	-/-	-/-
12	99.28/99.60	97.45/98.64	99.09/99.51	99.90/99.92	96.92/98.44	97.70/98.74	93.55/96.67	-/-	-/-
14	99.22/99.56	97.39/98.61	99.04/99.49	99.90/99.92	96.92/98.44	97.77/98.77	93.41/96.60	-/-	-/-
16	99.25/99.58	97.21/98.53	99.10/99.52	99.88/99.92	97.04/98.50	97.75/98.76	93.71/96.74	-/-	-/-
18	99.28/99.59	97.27/98.55	99.07/99.51	99.88/99.92	96.96/98.46	97.90/98.83	93.39/96.58	-/-	-/-
20	99.32/99.61	97.24/98.54	99.10/99.52	99.90/99.93	96.96/98.46	97.72/98.74	93.36/96.57	-/-	-/-

Interestingly, we find that P2P and QUIC traffic presented similar detection rates for ISP traces in spite of having quite dissimilar numbers of samples in the datasets (Table 2). The same happened for HOST traffic, DNS obtained high per-class metrics in spite of the fact that this class populated only the 9.12% and 7.21% of samples for HOST-1 and HOST-2. This fact indicates that the difficulty of detecting some kinds of application is not directly related to the class populations and there may exist other causes of performance degradation, such as: overlapping samples in the attribute space.

In order to compare the solutions to Class Imbalance in terms of performance increases or decreases with respect to the base estimator, we had to establish a baseline for each dataset. As this study is focused on Class Imbalance, we selected the models that produced the best results in terms of MAUC and/or GM to set the baselines. Thus, we have selected the model with 8 and 14 attributes for ISP-1 and ISP-2 respectively. We set the model with 18 attributes as baseline in the case of HOST-1, as it yielded the highest MAUC and OA. Finally, we chose the model including 12 predictors for HOST-2, since it produced the highest MAUC and BA accomplishing also the second best GM.

4.2 Addressing Class Imbalance: Algorithm comparison

In this section we present the comparison between the algorithms chosen to confront Class Imbalance in our NTC datasets. The comparison is firstly carried out in terms of global metrics, and per-class metrics are thoroughly explored for the most interesting techniques in Section 4.3. The results discussed correspond to the best-performing models in terms of MAUC, and they are presented as performance differences between each algorithm and the baselines set at Section 4.1. Firstly, we present the results obtained from experimenting with ISP traffic (Table 6) and secondly we focus on HOST network environment (Table 7). Finally, we statistically validate the results and present general remarks about the outcomes at Section 4.2.3.

4.2.1 ISP Network Environment

Table 6 shows the results for ISP-1 and ISP-2. Regarding oversampling on ISP-1, we find that all the algorithms generally performed well improving the scores obtained by the baseline. The best-performing algorithm in terms of OA and BA was SMOTE-B1, which increased the baseline by 4.92% and 3.8% respectively, meanwhile SMOTE yielded the second highest OA and BA. If we observe MAUC and GM, ROS obtains the best scores overcoming the baseline in 5.05% and 9.48%. When ISP-2 was oversampled, we observe that ROS remained to be the best method in terms of MAUC and GM, with increases of 4.08% and 8.05%. However, the observations on OA and BA change comparing to ISP-1. In this instance, the highest OA and BA were yielded by ADASYNC, which boosted both metrics in more than 6%. Interestingly, SMOTE-B1, SMOTE-B2 and ADASYNC produced quite negative impacts on MAUC and GM, evidencing that they did not clearly solve Class Imbalance for ISP-2. As the differences in performances between ISP traces reveal, the ISP-2 imposed a more difficult challenge than ISP-1 for oversampling. Note also that the size increase for ISP-1 was larger than ISP-2 due to the fact that ISP-2 present two minority classes more than ISP-1 (see Table 2).

When undersampling techniques were employed on ISP-1, TL obtained the best MAUC and GM with increases of 5.08% and 9.53% nearly followed by ENN, NCR and OSS. These algorithms also obtained the highest OAs and BAs amongst all the undersampling techniques, and ENN and NCR exactly yielded the same results for all global metrics. Note also that TL, ENN, NCR and OSS removed a low number of samples compared to other approaches. Other algorithms that notably overcame the baseline in terms of MAUC and GM were RUS and IHT, but getting weaker increases. In the case of RUS, these improvements were coupled with loose OA and BA increases and with a considerable training subset size reduction (more than 60% of samples were removed). Unlike RUS, IHT did not achieve improvements in terms of MAUC and GM. Furthermore, we find that there are some algorithms that dramatically worsened all global metrics evidencing that they are not recommendable choices for this network trace, they are: NM-1, NM-2, NM-3 and CNN. The abrupt performance decays are due to the fact that these algorithms removed a significant number of instances leading to important information losses (CNN and NM-3 removed more than 90% of the original samples). In the instance of ISP-2, the bad results obtained by NM-1, NM-2, NM-3 and CNN confirm the detrimental effect of these algorithms for ISP traffic. These techniques strongly lessened all global metrics, being the decrease more abrupt for OA and BA metrics. The best-performing algorithms were NCR, ENN and TL when ISP-2 was undersampled. NCR and ENN anew obtained pretty similar global metrics with increases close to 1.8% for OA and BA, and increases of 4.1% and 8.11% for MAUC and GM respectively. In the case of IHT, we observe that MAUC and GM metrics were reinforced, but it also yielded important losses in terms of OA and BA. In the case of OSS and RUS, they significantly overcame the baseline in terms of MAUC and GM, however they got weak enhancements for OA and BA. The main difference between both techniques is that RUS notably reduced the size of the training dataset, meanwhile OSS only removed the 1.49% of samples. Similarly to oversampling, ISP-2 poses a greater challenge than ISP-1 for undersampling algorithms.

When hybrid techniques are applied to ISP-1, we find that all algorithms overcame the baseline for all global metrics. Among all the hybrid algorithms, SMOTE-TL yielded the highest MAUC and GM with increases of 4.56% and 8.43% respectively, so that it is the best hybrid method at confronting Class Imbalance for ISP-1. Additionally, SMOTE-B1-TL and SMOTE-B2-TL achieved also really positive results, meanwhile the methods that combine SMOTE and ENN provided very weak improvements for MAUC and GM. While on OA and BA, we observe from Table 6 that SMOTE-B1-TL improved the baseline in 4.64% and 3.88% respectively, being the best-performing for these metrics. Another hybrid techniques that notably increased OA and BA were SMOTE-ENN, SMOTE-TL and SMOTE-B2-TL. Conversely, the slightest increases in terms of OA and BA were exhibited by SMOTE-B2-ENN and SMOTE-B1-ENN. In the case of ISP-2, SMOTE-TL is anew the technique that most improved the baseline in terms of MAUC and GM, it increased MAUC by 3.29% and GM by 6.39%. SMOTE-B1-TL, SMOTE-B2-TL and SMOTE-ENN also outperformed the baseline for MAUC and GM, but their enhancements were not as significant as SMOTE-TL. Focusing on OA and BA, the best OA and BA were obtained by SMOTE-B1-ENN followed by SMOTE-ENN, however the former negatively affected MAUC and GM. In general, all hybrid algorithms produced positive outcomes for all global metrics

but, on the contrary, SMOTE-B1-ENN and SMOTE-B2-ENN worsened MAUC and GM. In the case of applying hybrid approaches to ISP traces, these techniques also provided better results for ISP-1 than ISP-2.

In the case of training ensemble algorithms with ISP-1, RUSboost and TLboost tied for MAUC and GM yielding the highest enhancements with increases of 5.02% and 9.88% respectively. Furthermore, EE also obtained pretty relevant increases according to MAUC and GM, being the third scored ensemble method. Generally, all ensemble techniques provided quite remarkable enhancements for these metrics, achieving also important increases for OA and BA in specific cases. That is the case of ROSboost and SMOTEboost, which yielded quite beneficial results for all global metrics accomplishing the two highest OAs and BAs amongst all ensemble techniques. According to these performance metrics, the rest of algorithms did not achieve results as significant as ROSboost and SMOTEboost, and even BC loosely underperformed the baseline in terms of BA. Focusing on ISP-2, we observe similar outcomes to the ISP-1. The best ensemble algorithms at dealing with Class Imbalance for ISP-2 were RUSboost, EE and TLboost achieving increases superior to 4% for MAUC and superior to 8% for GM. The rest of algorithms also got positive outcomes for these metrics, however they were inferior to the former techniques. Regarding OA and BA, we find that ROSboost and SMOTEboost obtained the highest performances incrementing OA in more than 8.1% and in more than 7.5% respectively. Although the other techniques did not perform as well as ROSboost and SMOTEboost, they also overcame the baseline in terms of OA and BA with the exception of BC. Surprisingly, ensemble algorithms yielded higher enhancements for ISP-2 than ISP-1 in contrast to the data-levels algorithm previously discussed.

When MetaCOST was employed on ISP-1, we observe that it achieved to compensate Class Imbalance improving MAUC and GM in 4.41% and 9.13% respectively. On the contrary, MetaCOST weakened OA and BA with decreases of -1.48% and -1.19%. The same happened when MetaCOST was used to apply cost-sensitive to ISP-2, MAUC and GM were greatly strengthened, in contrast to OA and BA that deteriorated. In this case, the improvements on ISP-1 were more significant than ISP-2.

Table 6. Global metrics obtained for ISP network environment. The results are expressed as percentage increments or decrements respecting with the baseline

	ISP-1						ISP-2					
	OA	**BA**	**MAUC**	**GM**	**%**	**#F**	**OA**	**BA**	**MAUC**	**GM**	**%**	**#F**
OVERSAMPLING												
ROS	3.31	2.53	**5.05**	9.48	335.75	16	1.69	1.98	4.08	8.05	461.65	18
SMOTE	4.01	3.26	4.55	8.41	335.75	16	2.41	2.58	3.16	6.15	461.65	20
SMOTE-B1	**4.92**	**3.84**	3.48	6.10	335.75	18	2.26	2.37	-2.84	-6.88	461.65	14
SMOTE-B2	3.57	2.84	3.23	5.84	335.75	18	1.45	1.06	-2.55	-6.26	461.65	16
ADASYNC	3.41	2.82	0.57	0.53	336.18	12	**6.77**	6.19	-2.61	-6.92	461.74	18
UNDERSAMPLING												
RUS	2.1	1.14	4.82	9.23	-60.25	20	0.16	0.52	3.85	7.79	-67.08	18
CNN	-38.25	-39.17	-5.46	-7.35	-91.62	10	-63.6	-61.07	-8.46	-19.69	-91.29	10
TL	3.38	2.29	5.08	9.53	-0.73	18	1.44	1.42	4.06	-0.55	8.04	18
NM-1	-36.34	-38.73	-4.01	-4.58	-60.25	20	-52.02	-52.16	-5.76	-10.76	-67.08	12
NM-2	-50.42	-53.29	-6.39	-11.32	-60.25	20	-55.78	-56.24	-5.95	-12.5	-67.08	16
NM-3	-61.96	-62.09	-11.32	-21.93	-92.8	8	-72.95	-72.66	-10.28	-31.22	-91.91	8
OSS	2.99	2.04	4.99	9.42	-1.63	16	0.5	0.45	3.83	-1.49	7.68	10
ENN	3.09	2.24	5	9.43	-2.44	16	1.78	1.82	4.1	8.11	-3.8	20
NCR	3.09	2.24	5	9.43	-3.17	18	1.79	1.84	4.1	8.11	-3.8	20
IHT	-5.35	-5.93	3.44	7.42	-16.41	16	-11.25	-10.38	2.02	4.73	-27.75	20
HYBRID SAMPLING												
SMOTE-TL	3.95	3.29	4.56	8.43	57.61	18	2.55	2.59	3.29	6.39	64.48	16
SMOTE-B1-TL	4.64	3.88	4.06	7.31	58.08	20	3.83	3.38	2.11	3.68	65.22	18
SMOTE-B2-TL	3.91	2.99	3.86	7.04	53.09	20	3.62	2.95	1.78	3.04	58.63	18
SMOTE-ENN	4.05	3.1	2.9	5.22	31.57	20	4.5	4.08	0.78	1.17	34.55	14
SMOTE-B1-ENN	3.51	2.77	2.39	4.23	40.65	16	5.13	4.9	-0.09	-1.14	42.76	14
SMOTE-B2-ENN	2.78	2.07	1.25	2.04	14.81	10	2.51	2.1	-2.4	-5.98	11.67	12
ENSEMBLE ALGORITHMS												
EE	0.9	1.18	4.96	9.87	-	20	1.62	1.89	4.12	8.14	-	18
BC	0.29	-0.03	4.69	9.46	-	18	-0.01	-0.43	3.84	7.78	-	18
ROSboost	5.48	5.22	5	9.21	-	16	8.16	7.56	3.52	6.12	-	18
SMOTEboost	5.6	5.48	4.59	8.38	-	16	8.11	7.67	3.02	5.06	-	16
RUSboost	1.7	1.61	5.02	9.88	-	18	2.29	2.65	4.21	8.23	-	20
TLboost	1.99	1.6	5.05	9.88	-	18	1.56	1.9	4.1	8.11	-	20
COST-SENSITIVE												
MetaCOST	-1.48	-1.19	4.41	9.13	-	16	-2.36	-2.2	3.42	7.04	-	16

4.2.2 HOST Network Environment

Table 7 contains the results obtained via applying the different techniques to solve Class Imbalance for HOST datasets. When oversampling techniques were applied to HOST-1, ROS and SMOTE produced the best MAUCs and GMs with increases exceeding 0.55% and 1% respectively, so that they are the two best oversampling methods at solving Class Imbalance for HOST-1. Although SMOTE-B1 & B2 and ADASYNC overcame the baseline for all global metrics, they provided weak increases for MAUC and GM compared to ROS and SMOTE. Focusing exclusively on OA and BA, we find that ADASYNC achieved the highest increases, 0.7% for OA and 0.63% for BA. In addition, ROS and SMOTE also yielded very remarkable improvements in terms of OA. When HOST-2 was oversampled, we find that ROS was anew the best method in terms of MAUC and GM,

increasing MAUC by 0.55% and MAUC by 1.07%. These increases were also accompanied by significant improvements in terms of OA and BA, being ROS the best-performing techniques for OA. Additionally, SMOTE and ADASYNC also overcame the baseline for OA and BA, and even ADASYNC provided the highest BA. In the instance of SMOTE, this algorithm accomplished the second best MAUC and GM followed by ADASYNC. Unlike other oversampling algorithms, SMOTE-B1 and SMOTE-B2 negatively affected the predictive power of the models decreasing all global metrics when they were applied to HOST-2. In this case, the outcomes obtained for HOST-2 were slightly poorer than HOST-1.

Table 7. Global metrics obtained for HOST network environment. The results are expressed as percentage increments or decrements respecting with the baseline

	HOST-1						HOST-2					
	OA	**BA**	**MAUC**	**GM**	**%**	**#F**	**OA**	**BA**	**MAUC**	**GM**	**%**	**#F**
OVERSAMPLING												
ROS	0.59	0.49	0.66	1.26	131.16	20	0.32	0.45	0.55	1.07	287.62	14
SMOTE	0.67	0.47	0.56	1.05	131.16	16	0.3	0.42	0.47	0.92	287.62	18
SMOTE-B1	0.10	0.58	0.10	0.18	131.16	14	-0.44	-0.23	-0.64	-1.31	287.62	14
SMOTE-B2	0.40	0.60	0.17	0.29	131.16	18	-0.91	-0.92	-0.85	-1.61	287.61	18
ADASYNC	0.70	0.63	0.23	0.36	130.87	18	0.3	0.54	0.13	0.21	287.55	18
UNDERSAMPLING												
RUS	0.38	0.40	0.60	1.16	-32.21	20	0.25	0.29	0.57	1.11	-41.14	12
CNN	-11.9	-4.99	-4.80	-8.52	-73.03	20	-2.77	-5.05	-1.78	-3.29	-69.52	12
TL	0.52	0.43	0.64	1.23	-0.21	20	0.26	0.3	0.57	1.11	-0.03	12
NM-1	-14.1	-14.85	-3.88	-6.41	-32.21	20	0.26	0.3	0.56	1.09	-41.14	20
NM-2	-21.23	-5.31	-6.36	-11.09	-32.21	12	0.12	0.19	0.37	0.75	-41.14	10
NM-3	-26.77	-14.98	-9.28	-15.97	-73.29	20	-9.35	-9.43	-5.83	-13.1	-69.65	12
OSS	0.10	0.37	0.49	0.99	-3.06	12	-1.73	-2.4	0.13	0.56	-45.06	16
ENN	0.36	0.47	0.60	1.16	-1	20	0.25	0.33	0.56	1.09	-0.15	14
NCR	0.36	0.47	0.60	1.16	-1	20	0.25	0.33	0.56	1.09	-0.15	14
IHT	-4.53	-3.07	-0.85	-1.08	-9.93	18	0.02	0.15	0.51	1.04	-1.02	12
HYBRID SAMPLING												
SMOTE-TL	0.6	0.38	0.53	0.99	31.44	16	0.25	0.25	0.46	0.90	40.86	12
SMOTE-B1-TL	-0.49	-0.21	-0.27	-0.52	30.36	18	-0.35	-0.23	-0.61	-1.24	40.29	18
SMOTE-B2-TL	-0.26	-0.18	-0.24	-0.51	27.84	18	-0.2	-0.12	-0.34	-0.72	36.55	20
SMOTE-ENN	0.34	0.5	-0.03	-0.09	26.49	20	-0.19	-0.11	-0.09	-0.15	37.35	10
SMOTE-B1-ENN	-0.24	0.55	-0.18	-0.33	15.86	18	-0.69	-0.46	-1.04	-2.11	34.12	20
SMOTE-B2-ENN	-0.46	0.42	-0.51	-0.99	11.07	18	-0.52	-0.2	-1.06	-2.24	20.56	18
ENSEMBLE ALGORITHMS												
EE	0.55	-0.01	0.65	1.23	-	16	0.28	0.37	0.58	1.12	-	12
BC	0.33	-0.25	0.58	1.12	-	16	0.28	0.37	0.58	1.12	-	12
ROSboost	0.5	-1.23	0.47	0.89	-	18	0.17	0.41	0.52	1.02	-	12
SMOTEboost	0.49	0.23	0.44	0.82	-	18	-0.27	-0.21	0.18	0.43	-	14
RUSboost	0.17	-0.29	0.54	1.06	-	20	0.24	0.32	0.56	1.10	-	12
TLboost	0.54	0.06	0.65	1.23	-	20	0.3	0.38	0.58	1.12	-	12
COST-SENSITIVE												
MetaCOST	0.46	0.07	0.6	1.13	-	18	0.26	0.36	0.54	1.04	-	14

When HOST-1 was undersampled, we find from Table 7 that TL is the best algorithm at confronting Class Imbalance for this dataset, improving the baseline in 0.64% for MAUC and 1.23 for GM. Additionally, RUS, ENN and NCR also achieved positive results obtaining the same performances in terms of BA, MAUC and GM overcoming the baseline in 0.47%, 0.6% and 1.16% respectively. While on OA, TL got the highest OA with an increase of 0.52%, and RUS slightly outperformed ENN and NCR. Another algorithm that more loosely overcame the baseline for all global metrics was OSS, but its enhancements are not as remarkable as the former techniques. As it happened for ISP datasets (Table 6), CNN, NM-1, NM-2 and NM-3 had huge negative impacts on HOST-1. Surprisingly, IHT did not achieve overcoming the baseline for any metrics explored in contrast to ISP datasets. In the case of undersampling HOST-2, RUS and TL provided the highest increases in terms of MAUC and GM, nearly followed by NM-1, ENN and NCR. The main differences between the algorithms RUS, NM-1 and TL, ENN, NCR is the sample reduction rate, since the former techniques removed more than 40% of samples and the latter less than 0.20%. Among all undersampling techniques, the highest OAs were obtained by TL and NM-1; meanwhile, ENN and NCR outperformed the rest of algorithms for BA. Another algorithms that improved all global metrics comparing to the baseline were NM-2 and IHT. Surprisingly, the performances exhibited by NM-1 and NM-2 on HOST-2 notably differ from the observed for the rest of datasets, in this case all global metrics were reinforced. Observing the outcomes provided by OSS, we find that OA and BA were worsened comparing to baseline, in contrast to MAUC and GM that were loosely strengthened. Unlike for other datasets, the only two undersampling algorithms that reported negative impacts on all global metrics were CNN and NM-3.

When hybrid sampling was applied to HOST-1, the best-performing technique to confront Class Imbalance was SMOTE-TL according to MAUC and GM. This method was the only hybrid approach that enhanced all global metrics with respect to the baseline. Additionally, SMOTE-ENN also increased some performance metrics comparing to baseline, specifically the metrics that are sensitive to Class Imbalance (OA and BA). The highest BA was accomplished by SMOTE-B1-ENN, which accurately classified 0.55% of bytes more than the base estimator. Combining SMOTE-B1 or B2 with TL or ENN leaded to performance degradations with the exception of BA for SMOTE-B1-ENN and SMOTE-B2-ENN. Focusing on HOST-2, we find that the only hybrid algorithm that overcame the baseline for all global metrics was anew SMOTE-TL. This technique achieved increases of

0.25% for both OA and BA, and increases of 0.46% and 0.90% for MAUC and GM respectively. The rest of approaches obtained negative results for all global metrics when they are employed on HOST-2. The most unsatisfactory results in terms of OA and BA were obtained by SMOTE-B1-ENN, whereas SMOTE-B2-ENN yielded the poorest MAUC and GM with decreases of -1.06% and -2.11% respectively.

As Table 7 shows, ensemble algorithms that include resampling while learning comprise interesting solutions to deal with Class Imbalance. When these algorithms were trained with HOST-1, the best results in terms of MAUC and GM were achieved by EE and TLboost, which increased MAUC by 0.65% and GM by 1.23%. All ensemble algorithms outperformed the baseline for these metrics, and namely that BC and RUSboost obtained also very positive result. While on OA, EE obtained the highest score overcoming TLboost slightly, in contrast to BA for which the latter improved the baseline and the former underperformed it. The highest BA was obtained by SMOTEboost with an increase of 0.23%, and the rest of ensemble algorithms yielded BA decays with the exception of TLboost. Namely, ROSboost decreased BA with respect to the baseline by -1.23%. When ensemble algorithms were employed on HOST-2, we find that three algorithm tied in terms of MAUC and GM. EE, BC and TLboost obtained the best results for these performance metrics improving the baseline by 0.58% for MAUC and 1.12% for GM. Furthermore, the rest of algorithms also overcame the baseline for MAUC and GM achieving positive results, especially RUSboost and ROSboost. Regarding OA, TLboost yielded the best outcomes increasing the baseline by 0.3%, nearly followed by EE, BC and RUSboost. Among all the six ensemble algorithms, ROSboost yielded the best results in terms of BA, and other algorithms that produced positive results for this metric are: TLboost, EE, BC and RUSboost. Furthermore, ROSboost also improved the baseline in terms of BA, whereas OA and BA deteriorated when SMOTEboost was applied to HOST-2.

In the case of the cost-sensitive approach studied, we observe from Table 7 that MetaCOST improved all global metrics for both dataset (HOST-1 and HOST-2). When MetaCOST was applied to HOST-1, we find that OA and BA increased by 0.46% and 0.06% respectively; meanwhile, MAUC and GM improved in 0.6% and 1.13%. In the case of HOST-2, the performance increases were loosely lower than for HOST-1 with the exception of BA, which increased by 0.36%.

4.2.3 Statistical Validation & General Remarks

In the previous section we compared different type of solutions to Class Imbalance discussing their strengths and weakness in terms of all global metrics for the best models after applying FS. Through this section we pretend to confirm the previous observations validating statistically the results and to discuss more general remarks about the analyzed techniques. Table 8 contains the outcomes from applying the statistical approach presented at Section 3.3.4, which enables algorithm comparison against different datasets.

From Table 8, we find that some algorithms are fairly discarded as suitable solutions to confront Class Imbalance for our NTC dataset. The Friedman's scores obtained by these techniques are quite high revealing that they do not provide benefits for any global metric, or even they produced detrimental performances. These algorithms are: NM-3, CNN, NM-2, NM-1, IHT, OSS, SMOTE-B2, SMOTE-B1, SMOTE-B1-ENN and SMOTE-B2-ENN.

Other algorithms achieved reinforce metrics insensitive to imbalanced class distributions (MAUC and GM), but also they yielded very weak enhancements in terms of OA and BA. For example, the ensemble algorithms EE, BC, RUSboost and TLboost produced great improvements for MAUC and GM, and even TLboost and EE were the two best scored algorithms for these metrics. On the contrary, they obtained poor Friedman's scores for OA and BA. In addition, the data-level algorithms RUS, TL, ENN and NCR, SMOTE-TL, SMOTE-B1-TL, SMOTE-B2-TL and SMOTE-ENN also provided positive results for metrics insensitive to Class Imbalance. Note that ENN and NCR obtained the same Friedman's scores and that they were the best undersampling methods at mitigating Class Imbalance for our NTC datasets. Interestingly, we find that the best-performing techniques that employ undersampling tended to improve MAUC and GM notably, meanwhile they did not obtained so optimistic outcomes for OA and BA. In the case of MetaCOST, it did not obtained remarkable results for any of all the global metrics.

When ROSboost, SMOTEBoost, ADASYNC, ROS were applied to our datasets, we find that they notably strengthened OA and BA. Whereas they did not get so positive increases in terms of MAUC and GM. Among all the algorithms studied, ADASYNC was fairly the best-performing in terms of OA and BA for our datasets followed SMOTE. However, they did not yield so significant improvements for MAUC and GM. While on ROS, it achieved to improve all global metrics preserving a quite interesting tradeoff among metrics that are sensitive to Class Imbalance and the metrics that are not. ROSboost was the best ranked ensemble algorithm in terms of OA and BA followed by SMOTEboost, however they did not produce so notable improvements for the rest of metrics.

In short, the findings observed up to this point can be summarized in the following brief remarks:

- The algorithms that involve oversampling tend to reinforce the metrics that are sensitive to Class Imbalance (OA and BA). As we will show at Section 4.3, these improvements are directly related to increases in the individual accuracy of majority classes. Interestingly, ROS was able to provide benefits for both minority and majority traffic applications achieving quite positive outcomes in terms of GM and MAUC.
- The algorithms that include undersampling are prone to solve Class Imbalance and not to reinforce majority classes uniquely. Although some of them provided quite detrimental outcomes due to an excessive information removal, there are also some undersampling methods that constitute an interesting solution to imbalanced NTC. And particularly, RUS achieved to improve MAUC and GM with a significant sample reduction in spite of its simplicity, leading to faster training times.
- The Hybrid approaches considered did not provide significant benefits for imbalanced NTC comparing to other data-level approaches. And more specifically, the combination of SMOTE and TL generally outperformed the techniques that combine SMOTE with ENN.

- Regarding ensemble algorithms, we find that some of them confronted Class Imbalance effectively. EE jointly with the methods that included undersampling with boosting (TLboost and RUSboost) notably improved MAUC and GM, and oppositely the methods combining oversampling and boosting were prone to boost OA and BA more clearly than MAUC and GM.
- The cost-sensitive approach assumed achieved to increase MAUC and GM, however it produced losses in terms of OA and BA. However, further experimentation could be performed to study other more effective ways for computing classification costs.
- Through the experimentation on different datasets extracted from two network scenarios presenting quite dissimilar conditions, we find that some techniques present a more stable behavior than others. A clear example of a stable technique is TLboost, which performed uniformly on the different datasets. In the opposite side we find SMOTE-B1 and OSS, which produced quite dissimilar outcomes for different datasets.
- Accordingly to the metrics explored in our experiments, we find quite interesting to assess global metrics that are sensitive to Class Imbalance jointly to metrics that are not. As we have probed in previous sections, tradeoffs between performance metrics could exist and monitoring several of them is highly recommendable.
- Finally, network environments could present different Class Imbalance properties among them. In our work, the ISP environment constitutes the most challenging network scenario presenting a higher level of Class Imbalance. Interestingly, we find that performance losses are not exclusively related to class distributions, so that poor accuracies could also be related to other facts, such as: packet losses, packets out of order, overlapping regions and/or outliers.

In the following section we pretend to analyze individual accuracies for majority and minority classes. We focus the discussion on the most interesting methods explored with the purpose of validating their outcomes for the most challenging NTC dataset.

Table 8. Friedman´s Test. R_j denotes the scores obtained by each algorithm

	OA	BA	MAUC	GM
OVERSAMPLING				
ROS	8.25	8.25	4.87	5.50
SMOTE	5.08	6.16	13.00	12.50
SMOTE-B1	13.87	10.12	20.75	20.75
SMOTE-B2	15.75	13.75	20.75	20.75
ADASYNC	4.33	3.75	20.37	20.75
UNDERSAMPLING				
RUS	14.31	16.75	6.56	6.45
CNN	26.25	26.25	26.50	26.50
TL	10.91	13.37	4.12	7.20
NM-1	20.41	22.37	19.81	20.16
NM-2	24.00	23.75	24.00	24.00
NM-3	28.00	28.00	28.00	28.00
OSS	20.12	19.50	12.37	14.62
ENN	12.06	11.66	3.83	5.12
NCR	11.81	11.41	3.83	5.12
IHT	22.25	22.25	18.25	16.37
HYBRID SAMPLING				
SMOTE-TL	6.06	10.50	13.25	12.62
SMOTE-B1-TL	13.50	12.87	19.00	19.00
SMOTE-B2-TL	13.75	13.25	19.25	19.00
SMOTE-ENN	10.75	8.75	19.75	19.25
SMOTE-B1-ENN	14.50	10.75	21.75	21.50
SMOTE-B2-ENN	17.75	14.75	23.25	23.25
ENSEMBLE ALGORITHMS				
EE	11.87	14.87	3.79	2.25
BC	16.37	17.87	7.66	6.16
ROSboost	6.50	7.75	10.33	12.50
SMOTEboost	8.00	9.50	14.75	15.25
RUSboost	15.50	14.75	5.06	4.62
TLboost	11.08	13.75	2.41	2.12
COST-SENSITIVE				
MetaCOST	15.41	17.50	10.43	10.12
p-value	0.0015	0.0011	<0.0001	<0.0001

4.3 Analysis of per-class metrics

Up to this point, a wide number of solutions to Class Imbalance were compared analyzing their strengths and weaknesses in terms of global metrics. We found that the effectiveness of each technique depends on the metrics observed and also on the network environments. Through this section, we analyze in more detail the ability of reinforcement minority classes for some

algorithms aiming to confirm the suitability of them to be applied to imbalanced NTC. In order to not collapse the article with redundant results, we focus uniquely on the most remarkable algorithms and the most challenging dataset according to the results previously discussed. As aforementioned, ISP-2 constitutes the most challenging dataset, thus we report the per-class metrics obtained for this dataset. Regarding the algorithms discussed in this section, we have selected at least one algorithm from each approach considered. While on oversampling techniques, ROS has been selected due to the fact that it is the best-performing oversampling method in terms of MAUC and GM. Additionally, ADASYNC obtained the best Friedman's scores for OA and BA between all the algorithms studied, and also it has been included in this section. NCR and SMOTE-TL are also studied, since they obtained the highest MAUC and GM for their respective resampling approaches according to Table 8. Regarding ensemble algorithms, as TLboost was the most remarkable method between all the comparison algorithms, we have selected it for this section. Finally, we have included MetaCOST. Thereby, Table 9 contains the per-class accuracies obtained over ISP-2, the results are presented as increases or decreases comparing to the best model produced by the base estimator. As useful information for the subsequent discussion, we remember that the best-performing and that the minority classes for this dataset are (Table 4): P2P, INT, S/C, MEDIA, E/C and QUIC.

Regarding the metrics exhibited by ROS, per-class enhancements were not so positive when six or less predictors were chosen. This fact could likely be caused by the low predictive power of these subsets, since these subset sizes also produced negative outcomes when the base estimator was trained (Table 4). Although the best model in terms of MAUC was produced with 18 attributes (Table 7), significant enhancements on per-class metrics were observed with less features. For example, when models with more than eight predictors were selected, we find that the most of classes benefit from applying this oversampling technique. In general, the performance improvements of minority classes were very significant, and even the majority classes were also strengthened with the exception of DNS for specific subset sizes. Namely, ROS increased ACC and AUC for MEDIA (which was the most punished class by the base estimator, see Table 5) by more than 20% and 10% respectively and, similarly, E/C got important performance increases.

While on ADASYNC, we find that all minority classes were negatively affected for all subset sizes studied, being P2P the most damaged class with decreases that reached -47.89% and -23.47% for ACC and AUC respectively. On the contrary, WWW and DNS metrics were notably improved accomplishing the most significant increases for these classes between all the algorithms discussed through this section. Specifically, ACCs for WWW and DNS were increased by more than 7% and 10% when more than 10 attributes were selected. Due to this fact, ADASYNC obtained the best results in terms of OA and BA, meanwhile it exhibited quite detrimental performances for GM and MAUC.

Something similar to ROS happened when NCR was applied to undersample ISP-2, no evident improvements were observed on all classes when subset sizes equal or lesser than six were selected. In the case of selecting six predictors, some classes were strengthened, however the most of them were significantly punished. After that point, almost all per-class performances increased with the exception of WWW and DNS for certain subset sizes. The classes that exhibited the worst performances for the baseline were significantly improved, but with weaker increases than ROS. Conversely, other minority classes exhibited greater performances than using ROS, which contributed to the fact that NCR achieved better MAUCs and GMs than ROS, on contrast to OA and BA. The best model from employing NCR on ISP-2 were produced with 20 features, obtaining notable increases for all classes with the exception of DNS whose metrics were slightly worsened.

Regarding SMOTE-TL and similarly to ROS and NCR, we find that the most per-class metrics were worsened when less than eight attributes were selected. After that point, SMOTE-TL exhibited inferior improvements on minority classes to ROS and NCR, however the enhancements were also quite remarkable. While on majority classes, both WWW and DNS were reinforced with increases greater than 2.1% and 1.1% for their ACCs and AUCs. The performance increases exhibited on majority classes leaded SMOTE-TL to get better scores for OA than ROS and NCR (Table 8), but without reaching as significant increases as ADASYNC.

Among all the comparison algorithms, the best method at solving Class Imbalance was the ensemble technique TLboost, which is an original contribution of this work. Although significant increases on the most classes were observed for subset sizes greater than six, the best model was produced using 20 attributes. Note that per-class metrics for this subset size were generally greater than the obtained by ROS and NCR, with the exception of WWW, E/C and QUIC traffic.

Focusing on MetaCOST, we find a pretty different behavior from the previous algorithms. We find that majority classes are dramatically worsened comparing to baseline for all subset sizes considered, meanwhile minority classes were significantly improved when more than six predictors were selected. There are essentially one likely cause for this fact, remember that MetaCOST uses post-probability estimates and applies classification cost for relabeling the original training set. We experimented with several functions to compute classification costs, and finally the costs were computed according to Equation 9. The penalty on majority classes is strongly dependent on the cost computation, so that more optimal cost could conduct to better performance for MetaCOST. Finally, note that MetaCOST obtained the highest improvements on most of the minority classes amongst all methods discussed in this section, being the best model at improving QUIC, INT and S/C. Conversely, MEDIA, E/C and P2P obtained similar increases to TLboost.

Table 9. Per-class metrics produced by the selected techniques on ISP-2. The baseline corresponds with the model formed by 14 features

		P2P	WWW	DNS	INT	S/C	MEDIA	E/C	QUIC
		ACC/AUC	ACC/AUC	ACC/AUC	ACC/AUC	ACC/AUC	ACC/AUC	ACC/AUC	ACC/AUC
ROS									
	2	4.42/1.74	-57.11/-27.83	-15.23/-8.94	8.73/-14.2	2.28/0.43	-28.39/-14.4	-30.19/-14.9	-3.10/-3.41
	4	7.05/3.16	-45.77/-22.41	-6.08/-4.31	-1.06/-12.51	4.07/1.74	-15.68/-11.11	-26.76/-13.04	1.46/-0.12
	6	6.60/3.06	-32.04/-15.15	-4.48/-3.51	8.32/-6.35	4.28/2.03	-10.03/-5.35	-13.68/-6.54	1.84/0.57
	8	7.18/3.56	0.89/1.19	-0.68/-0.28	7.09/3.79	5.75/2.88	21.10/10.57	14.96/7.50	3.16/1.58
	10	7.11/3.52	0.47/1.07	-0.22/-0.02	7.33/3.87	5.85/2.94	21.22/10.57	15.64/7.79	3.56/1.81
	12	6.79/3.38	0.51/1.08	-0.03/0.07	7.24/3.75	5.60/2.82	20.92/10.44	15.59/7.84	3.81/1.94
	14	6.41/3.21	0.46/1.01	0.83/0.51	7.27/3.80	5.80/2.91	20.37/10.07	15.47/7.89	3.78/1.92
	16	6.60/3.28	0.62/1.09	0.28/0.24	7.43/3.85	5.53/2.82	21.10/10.42	15.61/7.99	3.51/1.79
	18	6.60/3.29	1.39/1.51	0.26/0.21	7.64/4.12	5.66/2.85	21.10/10.58	15.94/8.14	3.83/1.95
	20	6.60/3.26	1.37/1.50	-0.25/-0.05	7.66/4.20	5.66/2.83	21.10/10.55	15.98/8.13	3.59/1.81
ADASYNC									
	2	-44.68/-22.14	-50.98/-26.9	2.91/-0.78	-69.55/-33.25	-15.27/-7.79	-4.19/-21.69	-51.74/-24.93	-29.64/-14.95
	4	-46.67/-22.97	-31.67/-17.82	4.60/0.38	-9.11/-16.28	-12.93/-6.95	-55.25/-26.81	-49.01/-23.21	-16.3/-8.12
	6	-14.62/-6.91	4.78/-3.61	6.38/1.57	-70.38/-33.66	-10.79/-5.34	-45.28/-21.93	-37.11/-17.43	-10.0/-4.78
	8	-36.86/-18.64	4.07/1.15	6.64/3.07	-17.07/-7.86	-8.19/-4.00	-25.71/-12.32	-9.73/-4.02	-7.26/-3.42
	10	-6.99/-3.12	6.26/1.92	10.82/5.17	-17.12/-7.48	-6.92/-3.35	-31.49/-15.08	-12.96/-5.53	-5.54/-2.45
	12	-12.89/-6.00	7.33/2.31	11.38/5.49	-18.47/-7.98	-6.21/-2.91	-20.48/-9.55	-15.06/-6.50	-5.72/-2.53
	14	-13.46/-6.29	7.14/2.34	11.83/5.70	-15.59/-6.50	-6.23/-2.91	-27.66/-13.28	-11.97/-4.91	-5.56/-2.38
	16	-13.78/-6.53	4.70/1.08	11.48/4.74	-22.25/-9.95	-6.33/-2.97	-24.49/-11.57	-15.36/-6.85	-5.08/-2.21
	18	-15.9/-7.51	7.54/2.81	10.04/4.83	-12.24/-4.93	-4.69/-2.34	-17.75/-8.10	-8.11/-2.97	-6.19/-2.71
	20	-47.89/-23.47	7.11/2.53	11.89/5.68	-14.20/-5.86	-8.12/-3.87	-22.12/-10.3	-7.01/-2.59	-6.08/-2.68
NCR									
	2	3.20/1.40	-51.11/-27.23	-22.19/-12.02	8.65/-14.19	1.82/0.55	-29.05/-14.29	-31.28/-15.43	-3.86/-3.16
	4	6.86/3.11	-46.10/-23.04	-9.03/-5.53	-0.57/-12.35	4.46/1.68	-15.92/-11.28	-26.61/-13.03	1.38/-0.26
	6	7.18/3.29	-31.01/-16.42	-10.98/-6.05	8.55/-6.30	4.30/1.99	-10.33/-5.47	-13.80/-6.67	2.37/0.67
	8	7.11/3.54	0.50/0.85	-0.73/-0.30	7.20/3.84	5.64/2.84	20.61/10.39	15.28/7.51	3.21/1.60
	10	7.18/3.55	-0.04/0.52	-1.19/-0.45	7.07/3.65	5.85/2.93	20.61/10.30	15.35/7.50	3.56/1.78
	12	6.86/3.42	-0.22/0.58	-0.14/0.03	7.48/3.74	5.62/2.83	20.61/10.31	15.57/7.70	3.85/1.94
	14	6.60/3.31	0.27/0.72	-0.53/-0.13	7.19/3.71	5.66/2.84	20.55/10.28	15.45/7.68	3.91/1.94
	16	6.73/3.36	0.71/0.94	-0.61/-0.21	7.37/3.86	5.83/2.93	21.10/10.55	15.63/7.85	3.41/1.74
	18	6.73/3.36	-0.06/0.68	-0.63/-0.24	8.88/4.03	6.00/3.00	20.61/10.41	15.86/8.18	3.42/1.76
	20	7.18/3.56	1.68/1.43	-0.32/-0.03	7.84/4.21	6.21/3.11	20.73/10.48	15.83/8.12	3.81/1.94
SMOTE-TL									
	2	1.34/0.34	-54.93/-27.05	-12.95/-8.40	8.16/-14.44	-1.37/-1.06	-35.19/-17.54	-33.38/-16.38	-6.72/-4.62
	4	2.05/0.90	-44.26/-21.96	-3.82/-3.57	-1.56/-12.71	-0.39/-0.35	-25.22/-15.84	-31.03/-14.88	-3.25/-2.20
	6	3.14/1.48	-30.59/-14.75	-2.97/-2.85	7.87/-6.57	1.00/0.48	-18.78/-9.50	-17.24/-8.21	-0.19/-0.27
	8	4.42/2.26	1.52/1.35	1.78/0.92	5.29/2.94	3.30/1.74	16.36/8.34	10.90/5.56	2.38/1.24
	10	4.36/2.24	1.13/1.16	1.55/0.81	5.04/2.76	4.25/2.19	16.23/8.31	10.95/5.47	2.77/1.43
	12	4.04/2.08	1.09/1.08	1.71/0.92	5.03/2.66	4.01/2.09	15.14/7.79	10.95/5.49	2.90/1.52
	14	4.04/2.07	2.14/1.72	2.07/1.08	5.99/3.39	3.48/1.83	15.99/8.10	12.50/6.57	2.86/1.49
	16	4.17/2.14	2.23/1.77	2.15/1.12	6.18/3.47	3.69/1.94	15.02/7.64	12.845/6.75	2.83/1.48
	18	4.23/2.16	2.04/1.65	2.11/1.10	5.82/3.29	4.10/2.14	15.26/7.71	12.59/6.60	2.73/1.44
	20	4.36/2.24	2.12/1.69	2.67/1.38	5.69/3.25	4.00/2.08	14.84/7.60	12.65/6.60	2.71/1.42
TLboost									
	2	4.16/1.70	-57.16/-27.86	-15.04/-8.92	8.74/-14.20	2.16/0.43	-28.26/-14.41	-30.22/-14.92	-3.16/-3.45
	4	6.92/3.10	-45.80/-22.48	-6.26/-4.41	-0.97/-12.49	4.05/1.66	-15.74/-11.10	-26.60/-12.97	1.20/-0.23
	6	6.60/3.10	-31.90/-15.12	-4.69/-3.58	8.33/-6.32	4.21/1.99	-9.97/-5.34	-13.73/-6.63	1.88/0.62
	8	7.11/3.54	0.78/1.11	-0.44/-0.15	7.04/3.77	5.49/2.77	21.16/10.60	15.16/7.54	3.11/1.58
	10	7.05/3.50	0.09/0.84	-0.29/-0.05	7.15/3.75	5.74/2.90	21.34/10.65	15.48/7.57	3.59/1.82
	12	6.79/3.39	0.35/0.96	-0.19/-0.01	7.20/3.77	5.75/2.88	20.91/10.47	15.59/7.71	3.81/1.93
	14	7.05/3.50	0.52/1.03	-0.18/0.00	7.13/3.69	5.85/2.94	21.10/10.53	15.52/7.81	3.66/1.87
	16	6.86/3.41	0.78/1.18	0.03/0.10	7.46/3.90	5.87/2.96	21.16/10.58	15.79/7.98	3.50/1.79
	18	6.86/3.41	1.25/1.41	0.28/0.22	7.68/4.13	5.89/2.96	21.16/10.62	15.88/8.07	3.45/1.78
	20	6.86/3.42	1.16/1.36	0.36/0.26	7.70/4.13	5.91/2.96	21.34/10.70	15.86/8.06	3.69/1.90
MetaCOST									
	2	3.46/0.56	-59.99/-29.07	-22.76/-12.43	-63.89/-30.77	2.16/-0.08	13.44/-14.02	-29.87/-15.26	-4.84/-4.38
	4	6.28/2.61	-54.93/-26.55	-13.90/-7.99	-63.44/-30.38	5.17/1.81	18.12/-11.06	-26.31/-13.13	3.43/0.53
	6	5.77/2.40	-37.02/-17.53	-14.01/-7.79	-15.09/-14.15	4.84/1.67	-0.97/-6.36	-13.64/-6.90	3.16/0.71
	8	6.15/2.97	-2.64/-0.33	-7.78/-3.74	7.71/3.27	6.53/2.95	20.24/9.87	14.99/7.13	4.86/2.15
	10	7.18/3.48	-3.41/-0.69	-7.41/-3.56	6.28/2.59	6.62/3.02	21.03/10.22	15.08/6.90	4.90/2.21
	12	6.60/3.21	-3.19/-0.57	-7.54/-3.62	6.86/2.90	6.66/3.06	21.16/10.23	15.37/7.13	5.64/2.56
	14	6.86/3.32	-2.20/-0.08	-7.05/-3.37	8.34/3.60	6.89/3.21	21.28/10.22	15.74/7.83	5.61/2.55
	16	6.79/3.31	-2.19/-0.06	-7.15/-3.42	8.39/3.61	6.98/3.21	21.34/10.31	15.66/7.77	5.78/2.63
	18	6.60/3.19	-1.76/0.14	-7.12/-3.41	8.34/3.68	6.69/3.09	20.98/10.14	15.77/7.87	5.41/2.45

20	5.83/2.77	-1.92/0.06	-7.70/-3.71	8.26/3.64	6.33/2.89	20.55/9.95	15.68/7.75	5.28/2.37

The observations provided through this section confirm trade-offs between metrics sensitive to Class Imbalance and other that are not. Some algorithms strengthened minority classes, and eventually, these performance increases were accompanied also with improvements on the majority classes. Other interesting observation is that most of the techniques obtaining positive outcomes for MAUC and GM using less predictors than the best models provided as baseline. This fact leads to attributes savings, which could be an interesting feature for fast early NTC.

5. Conclusions and Future Work

Through this paper, 28 techniques to solve Class Imbalanced were analyzed and compared for our NTC datasets. To the best of our knowledge, this work constitutes the first study that analyzes an important number of solutions to Class Imbalance for multiclass NTC. Previous works limited the analysis to few methods or faced the problem simplifying it to binary subproblems. Our algorithm comparison involved: 21 data-level solutions, six ensemble techniques and one cost-sensitive approach. The selected techniques were tested on two different network environments evaluating several performance metrics to find out the strengths and weakness of each method. Among the algorithms studied, we presented two boosting algorithms that include data-level methods during learning, they are: ROSboost and TLboost. Additionally, some algorithms had to be adapted to multiclass problems using our own strategies to adjust the required parameters (Section 3.4). We make publicly available all algorithms and strategies implemented at [64], and encourage other authors to test them in their respective research fields.

As result of our comparison, we find that many of the techniques explored are able to benefit traffic classification models compensating performance losses due to Class Imbalance. Regarding metrics sensitive to imbalanced class distributions, we find that methods involving oversampling provided substantial improvements, being the algorithms that involve ROS and SMOTE the most promising approaches. Conversely, the algorithms that employ undersampling produced the best improvements for metrics insensitive to Class Imbalance, being our algorithm TLboost the best-performing for these metrics. However, they leaded to weak enhancements for OA and BA, being RUS the only undersampling algorithm that keep an interesting tradeoff between metrics sensitive and insensitive to imbalanced traffic distributions. As it has been reported in our result section, hybrid resampling did not get so positive results comparing to other solutions, and the same happened for MetaCOST. Furthermore, we have confirmed that minority classes are significantly benefit from applying the most relevant algorithms and that important enhancements can be achieved using less features than the baseline. The latter fact could constitute an interesting advantage for fast early NTC.

In order to extend and improve the contributions provided here, several research lines are envisioned as future work. Although we have considered several algorithm-level and one cost-sensitive approaches, there exists novel algorithms based on decision trees that could provide interesting enhancements for Class Imbalance. The lack of implementations of these algorithms was the decisive fact to not include them for our experiments. With respect to the cost-sensitive approach studied, we found that it produced negative outcomes for majority classes, so that experimenting with more sophisticated ways to compute classification cost may lead to more optimistic improvements. Furthermore, the comparison carried out in this work may be extended to other emerging knowledge areas such as: IoT and Smart Cities. Finally, studying these solutions with a finer classification granularity might constitute also an interesting future research line.

Acknowledgments

This work has been partially funded by the Ministerio de Economía y Competitividad del Gobierno de España and the Fondo de Desarrollo Regional (FEDER) within the project "Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software, Ref: TIN2014-57991-C3-2-P", in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento. Additionally, we would like to thank the Broadband Communications Research Group belonging to UPC BarcelonaTech, especially Valentín Carela-Español for providing the network traces we have used in our work. Furthermore, we would like to thank the ISP for the real network traffic captures and the resources shared with us for this work. And finally, we would like to thank the reviewers of Neurocomputing for the feedback provided, which has been very useful to upgrade our manuscript.

References

[1] J. Khalife, A. Hajjar, and J. Diaz-verdejo, “A multilevel taxonomy and requirements for an optimal traf fi c-classi fi cation model,” no. January, pp. 101–120, 2014.

[2] A. Callado et al., “A survey on internet traffic identification,” IEEE Commun. Surv. Tutorials, vol. 11, no. 3, pp. 37–52, 2009.

[3] T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” IEEE Commun. Surv. Tutorials, vol. 10, no. 4, pp. 56–76, 2008.

[4] A. Dainotti, A. Pescape, and K. Claffy, “Issues and future directions in traffic classification,” IEEE Netw., vol. 26, no. 1, pp. 35–40, Jan. 2012.

[5] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, “Robust Network Traffic Classification,” IEEE/ACM Trans. Netw., vol. 23, no. 4, pp. 1257–1270, 2015.

[6] L. Peng, B. Yang, and Y. Chen, “Effective packet number for early stage internet traffic identification,” Neurocomputing, vol. 156, pp. 252–267, 2015.

[7] Z. Liu, R. Wang, M. Tao, and X. Cai, “A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion,” Neurocomputing, vol. 168, pp. 365–381, 2015.

[8] L. Peng, H. Zhang, Y. Chen, and B. Yang, “Imbalanced traffic identification using an imbalanced data gravitation-based classification model,” Comput. Commun., vol. 102, pp. 177–189, 2017.

[9] H. Shi, H. Li, D. Zhang, C. Cheng, and W. Wu, “Efficient and robust feature extraction and selection for traffic classification,” Comput. Networks, vol. 119, pp. 1–16, 2017.

[10] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. Hernández Callejo, “Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal,” Comput. Networks, vol. 127, pp. 68–80, Nov. 2017.

[11] CAIDA, “CoralReef Software Suite,” 1999. [Online]. Available: http://www.caida.org/tools/measurement/coralreef/. [Accessed: 06-Jun-2018].

[12] “IANA, List of assigned port numbers.” [Online]. Available: http://www.iana.org/assignments/port-numbers.

[13] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, “nDPI: Open-source high-speed deep packet inspection,” in 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 617–622.

[14] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, “Unsupervised traffic classification using flow statistical properties and IP packet payload,” J. Comput. Syst. Sci., vol. 79, no. 5, pp. 573–585, 2013.

[15] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, “Learning from class-imbalanced data: Review of methods and applications,” Expert Syst. Appl., vol. 73, pp. 220–239, 2017.

[16] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, “Effective detection of sophisticated online banking fraud on extremely imbalanced data,” World Wide Web, vol. 16, no. 4, pp. 449–475, 2013.

[17] Y. Wang, X. Li, and X. Ding, “Probabilistic framework of visual anomaly detection for unbalanced data,” Neurocomputing, vol. 201, pp. 12–18, Aug. 2016.

[18] S. Shilaskar, A. Ghatol, and P. Chatur, “Medical decision support system for extremely imbalanced datasets,” Inf. Sci. (Ny)., vol. 384, pp. 205–219, Apr. 2017.

[19] J. Erman, A. Mahanti, and M. Arlitt, “Byte me: a case for byte accuracy in traffic classification,” Proc. 3rd Annu. ACM …, pp. 35–37, 2007.

[20] T. Qin, L. Wang, Z. Liu, and X. Guan, “Robust application identification methods for P2P and VoIP traffic classification in backbone networks,” Knowledge-Based Syst., vol. 82, pp. 152–162, 2015.

[21] H. Wei and B. Sun, “BalancedBoost : A Hybrid Approach for Real-time Network Traffic Classification,” 2014.

[22] Q. Liu and Z. Liu, “A comparison of improving multi-class imbalance for internet traffic classification,” Inf. Syst. Front., vol. 16, no. 3, pp. 509–521, 2014.

[23] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, “Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases,” Neurocomputing, vol. 175, pp. 935–947, 2016.

[24] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, “An insight into classification with imbalanced data : Empirical results and current trends on using data intrinsic characteristics,” Inf. Sci. (Ny)., vol. 250, pp. 113–141, 2013.

[25] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE : Synthetic Minority Over-sampling Technique,” vol. 16, pp. 321–357, 2002.

[26] H. Han, W. Wang, and B. Mao, “Borderline-SMOTE : A New Over-Sampling Method in,” pp. 878–887, 2005.

[27] H. He, Y. Bai, E. A. Garcia, and S. Li, “ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning,” no. 3, pp. 1322–1328, 2008.

[28] P. Hart, “The condensed nearest neighbor rule (Corresp.),” IEEE Trans. Inf. Theory, vol. 14, no. 3, pp. 515–516, May 1968.

[29] J. Zhang and I. Mani, “kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction,” Work. Learn. from Imbalanced Datasets II ICML Washingt. DC 2003, pp. 42–48, 2003.

[30] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, “Addressing imbalance in multilabel classification: Measures and random resampling algorithms,” Neurocomputing, vol. 163, pp. 3–16, 2015.

[31] D. L. Wilson, “Asymptotic Properties of Nearest Neighbor Rules Using Edited Data,” IEEE Trans. Syst. Man Cybern., vol. 2, no. 3, pp. 408–421, 1972.

[32] J. Laurikkala, “Improving identification of difficult small classes by balancing class distribution,” Proc. 8th Conf. AI Med. Eur. Artif. Intell. Med., pp. 63–66, 2001.

[33] M. R. Smith, T. Martinez, and C. Giraud-Carrier, “An instance level analysis of data complexity,” Mach. Learn., vol. 95, no. 2, pp. 225–256, 2014.

[34] I. Tomek, “Two Modifications of CNN,” IEEE Trans. Syst. Man. Cybern., vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.

[35] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, “Balancing Training Data for Automated Annotation of Keywords: a Case Study,” Proc. Second Brazilian Work. Bioinforma., pp. 35–43, 2003.

[36] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, “A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data,” SIGKDD Explor. Newsl., vol. 6, no. 1, pp. 20–29, 2004.

[37] X. Liu, J. Wu, and Z. Zhou, “Exploratory Under-Sampling for Class-Imbalance Learning,” 2006.

[38] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and a. Napolitano, “RUSBoost: Improving classification performance when training data is skewed,” 2008 19th Int. Conf. Pattern Recognit., no. March 2016, pp. 8–11, 2008.

[39] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, “SMOTEBoost: Improving Prediction of the Minority Class in Boosting,” pp. 107–119, 2003.

[40] P. Domingos, “MetaCost: a general method for making classifiers cost-sensitive,” in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’99, 1999, pp. 155–164.

[41] S. Wang and X. Yao, “Multiclass Imbalance Problems : Analysis and Potential Solutions,” vol. 42, no. 4, pp. 1119–1130, 2012.

[42] A. Fernández, V. López, M. Galar, M. José, and F. Herrera, “Analysing the classification of imbalanced data-sets with multiple classes : Binarization techniques and ad-hoc approaches,” Knowledge-Based Syst., vol. 42, pp. 97–110, 2013.

[43] Z. H. Zhou and X. Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” IEEE Trans. Knowl. Data Eng., vol. 18, no. 1, pp. 63–77, 2006.

[44] T. Hastie and R. Tibshirani, “Classification by pairwise coupling,” Ann. Stat., vol. 26, no. 2, pp. 451–471, 1998.

[45] R. Ryan and A. Klautau, “In Defense of One-Vs-All Classification,” Notes, vol. 7, pp. 101–141, 2004.

[46] N. Japkowicz, “Assessment Metrics for Imbalanced Learning,” in Imbalanced Learning, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013, pp. 187–206.

[47] J. G. Moreno-Torres, J. A. Saez, and F. Herrera, “Study on the Impact of Partition-Induced Dataset Shift on K-Fold Cross-Validation,” {IEEE} Trans. Neural Networks Learn. Syst., vol. 23, no. 8, pp. 1304–1312, 2012.

[48] V. López, A. Fernández, and F. Herrera, “On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed,” Inf. Sci. (Ny)., vol. 257, pp. 1–13, 2014.

[49] L. Bernaille, R. Teixeira, and K. Salamatian, “Early application identification,” Proc. 2006 ACM Conex. Conf., p. 6:1--6:12, 2006.

[50] L. Bernaille, R. Teixeira, I. Akodjenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” ACM SIGCOMM Comput. Commun. Rev., vol. 36, no. 2, pp. 23–26, 2006.

[51] W. Li and A. W. Moore, “A Machine Learning Approach for Efficient Traffic Classification,” in 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007, pp. 310–317.

[52] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification,” ACM SIGCOMM Comput. Commun. Rev., vol. 36, no. 5, p. 5, Oct. 2006.

[53] M. Soysal and E. G. Schmidt, “Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison,” Perform. Eval., vol. 67, no. 6, pp. 451–467, Jun. 2010.

[54] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. Hernández Callejo, “Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal,” Comput. Networks, vol. 127, pp. 68–80, 2017.

[55] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” ACM SIGMETRICS Perform. Eval. Rev., vol. 33, no. 1, p. 50, Jun. 2005.

[56] T. Auld, A. W. Moore, and S. F. Gull, “Bayesian Neural Networks for Internet Traffic Classification,” IEEE Trans. Neural Networks, vol. 18, no. 1, pp. 223–239, Jan. 2007.

[57] A. Este, F. Gringoli, and L. Salgarelli, “Support Vector Machines for TCP traffic classification,” Comput. Networks, vol. 53, no. 14, pp. 2476–2490, Sep. 2009.

[58] D. M. Divakaran, L. Su, Y. S. Liau, and V. L. Vrizlynn, “SLIC: Self-Learning Intelligent Classifier for network traffic,” Comput. Networks, vol. 91, pp. 283–297, 2015.

[59] J. Camacho, P. Padilla, P. García-teodoro, and J. Díaz-verdejo, “A generalizable dynamic flow pairing method for traffic classification,” Comput. Networks, vol. 57, no. 14, pp. 2718–2732, 2013.

[60] L. Peng, B. Yang, Y. Chen, and A. Abraham, “Data gravitation based classification,” Inf. Sci. (Ny)., vol. 179, no. 6, pp. 809–819, Mar. 2009.

[61] S. Egea, A. Rego, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, “Intelligent IoT Traffic Classification Using Novel Search Strategy for Fast Based-Correlation Feature Selection in Industrial Environments,” IEEE Internet Things J., 2018.

[62] S. E. Gómez, “FCBF module,” 2018. [Online]. Available: https://github.com/SantiagoEG/FCBF_module. [Accessed: 23-May-2018].

[63] G. Lemaitre, F. Nogueira, and C. K. Aridas, “Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning,” CoRR, vol. abs/1609.0, pp. 1–5, 2016.

[64] Santiago Egea Gómez, “GitHub - SantiagoEG/ImbalancedMulticlass,” 2018. [Online]. Available: https://github.com/SantiagoEG/ImbalancedMulticlass/tree/master. [Accessed: 06-Jun-2018].

[65] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2012.

[66] J. Demšar, “Statistical Comparisons of Classifiers over Multiple Data Sets,” J. Mach. Learn. Res., vol. 7, pp. 1–30, 2006.

[67] V. Carela-Español, T. Bujlow, and P. Barlet-Ros, “Is Our Ground-Truth for Traffic Classification Reliable?,” 2014, pp. 98–108.

[68] “nDPI – ntop.” [Online]. Available: https://www.ntop.org/products/deep-packet-inspection/ndpi/. [Accessed: 15-Feb-2018].

[69] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, “Independent comparison of popular DPI tools for traffic classification,” Comput. Networks, vol. 76, pp. 75–89, 2015.

[70] A. W. Moore and K. Papagiannaki, “Toward the Accurate Identification of Network Applications,” Springer, Berlin, Heidelberg, 2005, pp. 41–54.

[71] A. Callado, J. Kelner, D. Sadok, C. Alberto Kamienski, and S. Fernandes, “Better network traffic identification through the independent combination of techniques,” J. Netw. Comput. Appl., vol. 33, no. 4, pp. 433–446, Jul. 2010.

[72] R. Jhonson, “imbalanced-algorithms,” 2017. [Online]. Available: https://github.com/dialnd/imbalanced-algorithms. [Accessed: 23-May-2018].