Universidad de Valladolid

PROGRAMA DE DOCTORADO EN TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

TESIS DOCTORAL:

APPLICATION OF ADVANCED MACHINE LEARNING TECHNIQUES TO EARLY NETWORK TRAFFIC CLASSIFICATION

Presentada por Santiago Egea Gómez para optar al grado de
Doctor/a por la Universidad de Valladolid

Dirigida por:
Belén Carro Martínez, Luis Hernández Callejo y Antonio Javier Sánchez Esguevillas

Acknowledgements

… to who inspired this research.

… to who unconditionally backed me during this long adventure.

… to who has contributed to this research.

… to my supervisors and the people involved in this challenging journey.

… to my true friends and genuine relatives.

… to who they are not already in this world, but always are with me.

… they know who they are.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADASYNC	ADAptive SYNthetic algorithm
AUC	Area Under Curve
BA	Byte Accuracy
BC	Balance Cascade
BoF	Bag of Flows
CART	Classification And Regression Tree
CIFE	Conditional Infomax Feature Extractio
CMIM	CoNditional Mutual Information Maximization
CNN	Condensed Nearest Neighbor
CV	Cross Validation
DBSCAN	Density-Based Spatial Clustering Application with Noise
DPI	Deep Packet Inspection
EE	Easy Ensemble
EFOA	Efficient Feature Optimization Approach
EL	Extreme Learning
ENN	Edited Nearest Neighbor
FCBF	Fast Correlation Based Filter
FCBF#	Fast Correlation Based Filter with a different search strategy
FCBFiP	Fast Correlation Based Filter in Pieces
FFT	Fast Fourier Transform
FS	Feature Selecion
FTP	File Transfer Protocol
GA	Genetic Algorithm
GM	Geometric Mean
GMM	Gaussian Mixture Model
GOA	Global Optimization Approach
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICAP	Interaction CAPping
IDGB	Imbalanced Data Gravitation Based Classifier
IHT	Instance Hardness Threshold
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
JMI	Joint Mutual Information
KELM	Kernel-based Extreme Learning Machine
LOA	Local Optimization Approach
MIFS	Mutual Interformation Feature Selection
MIM	Mutual Information Maximization
MRMR	Minimum-Redundancy Maximum-Relevance
NCR	Neighborhood Cleaning Rule
NM	Near Miss
OA	Overal Accuracy
OSS	One Sided Selection
P2P	Peer to Peer
PCA	Principal Component Analysis
PCAP	Packet CAPture
PNN	Probabilistic Neural Networks
ROC	Receiver Operating Characteristic
ROS	Random OverSampling
RUS	Random UnderSampling
SLIC	Self Learning Intelligent Classifier
SMOTE	Synthetic Minority Oversampling TEchnique
SSH	Secure SHel
SVM	Support Vector Machine
TCEV	Traffic Classifier based on Expanding Vectors
TCP	Transport Congestion Protocol
T-DTC	Tailored Decision Tree Chain
TL	Tomek Link
UDP	User Datagram Protocol
WMI	Weighted Mutual Information
WSU	Weighted Symmetrical Uncertainty

ABSTRACT

The fast-paced evolution of the Internet is drawing a complex context which imposes demanding requirements to assure end-to-end Quality of Service. The development of advanced intelligent approaches in networking is envisioning features that include autonomous resource allocation, fast reaction against unexpected network events and so on. Internet Network Traffic Classification constitutes a crucial source of information for Network Management, being decisive in assisting the emerging network control paradigms. Monitoring traffic flowing through network devices support tasks such as: network orchestration, traffic prioritization, network arbitration and cyberthreats detection, amongst others.

The traditional traffic classifiers became obsolete owing to the rapid Internet evolution. Port-based classifiers suffer from significant accuracy losses due to port masking, meanwhile Deep Packet Inspection approaches have severe user-privacy limitations. The advent of Machine Learning has propelled the application of advanced algorithms in diverse research areas, and some learning approaches have proved as an interesting alternative to the classic traffic classification approaches. Addressing Network Traffic Classification from a Machine Learning perspective implies numerous challenges demanding research efforts to achieve feasible classifiers. In this dissertation, we endeavor to formulate and solve important research questions in Machine-Learning-based Network Traffic Classification. As a result of numerous experiments, the knowledge provided in this research constitutes an engaging case of study in which network traffic data from two different environments are successfully collected, processed and modeled.

Firstly, we approached the Feature Extraction and Selection processes providing our own contributions. A Feature Extractor was designed to create Machine-Learning ready datasets from real traffic data, and a Feature Selection Filter based on fast correlation is proposed and tested in several classification datasets. Then, the original Network Traffic Classification datasets are reduced using our Selection Filter to provide efficient classification models. Many classification models based on CART Decision Trees were analyzed exhibiting excellent outcomes in identifying various Internet applications. The experiments presented in this research comprise a comparison amongst ensemble learning schemes, an exploratory study on Class Imbalance and solutions; and an analysis of IP-header predictors for early traffic classification. This thesis is presented in the form of compendium of JCR-indexed scientific manuscripts and, furthermore, one conference paper is included.

In the present work we study a wide number of learning approaches employing the most advance methodology in Machine Learning. As a result, we identify the strengths and weaknesses of these algorithms, providing our own solutions to overcome the observed limitations. Shortly, this thesis proves that Machine Learning offers interesting advanced techniques that open prominent prospects in Internet Network Traffic Classification.

RESUMEN

La imparable evolución de Internet está dibujando un complejo entorno que impone numerosos requisitos para asegurar la Calidad de Servicio. La aparición de nuevos paradigmas de red inteligentes proyecta en los dispositivos de red capacidades como: la gestión automática de los recursos de la red, la reacción instantánea ante eventos inesperados, etc. La Clasificación de Tráfico de Red constituye una fuente de información esencial para el Gestión de Redes, siendo manifiesta su gran importancia para los nuevos paradigmas de red. Monitorizar el tráfico que circula por ciertos dispositivos de red conlleva importantes beneficios para tareas como: la orquestación de redes, priorización de tráfico, arbitraje en la utilización de la red y detección de ciber amenazas, entre otros.

Los clasificadores de tráfico tradicionales han quedado obsoletos debido a la rápida evolución de Internet. Los clasificadores basados en puertos sufren importantes pérdidas de precisión debido al enmascaramiento de puertos, mientras que los enfoques de Inspección Profunda de Paquetes están limitados debido a problemas de privacidad. La aparición de técnicas de Aprendizaje Automático ha abierto nuevas posibilidades en numerosos problemas, y se ha demostrado que algunas de ellas son una alternativa interesante para la Clasificación de Tráfico de Internet. Abordar este problema usando Aprendizaje Automático implica numerosos retos de investigación con el fin de lograr clasificadores eficientes. Esta tesis trata de formular y responder preguntas de investigación cruciales en esta área de investigación. Como resultado de numerosos experimentos, el conocimiento aportado en este trabajo constituye un valioso caso de estudio en el que tráfico proveniente de distintos entornos de red es recolectado, procesado y modelado con éxito.

Para empezar, se abordó el problema de recolección y selección de atributos aportando nuestras propias soluciones. Se diseñó un Extractor de Atributos para generar conjuntos de datos de Aprendizaje Automático a partir de tráfico de red; también se presentó un algoritmo de Selección de Atributos basado en medidas de correlación, cuyo rendimiento fue probado en varios problemas de clasificación. A continuación, se utilizó dicho algoritmo para reducir el número de atributos en nuestros conjuntos de datos de Clasificación de Tráfico buscando mejorar la eficiencia de nuestros modelos predictivos. Más adelante, se analizaron numerosos modelos de clasificación basados en Árboles de Decisión, produciendo excelentes resultados en la identificación de tráfico de red. Los experimentos presentados en esta tesis incluyen un análisis de algoritmos ensamblados, un estudio del problema de Clases Desequilibradas y soluciones para éste; y un análisis de distintos parámetros extraídos de cabeceras IP usados como atributos. Esta tesis se presenta como un compendio de artículos científicos publicados en revistas JCR indexadas, además de un artículo de conferencia.

En la presente investigación, se han estudiado un amplio número de técnicas de aprendizaje asumiendo las metodologías más avanzadas en Aprendizaje Automático. Como resultado, se identificaron las ventajas y desventajas de estos algoritmos, aportando también nuestras propias soluciones para solucionar las limitaciones observadas. En resumen, esta tesis demuestra que las técnicas avanzadas en Aprendizaje Automático tienen unas prometedoras perspectivas en la Clasificación de Tráfico de Internet.

1. Introduction

The World of the Internet is composed of millions of users around the world, millions of server computers providing a wide range of online facilities and a global infrastructure that interconnects countless local networks. Internet networks require innovating their infrastructures and control mechanisms to provide support to emerging services continuously growing and evolving [1]. Currently, the Internet provides communication to numerous public and private institutions that are unceasingly transferring sensitive information about their customers, products and facilities. Therefore, a communication disruption might bring important information leakages and huge economic losses obstructing the activity of industries and even governments [2]. Furthermore, the control and surveillance of certain critical infrastructures (such as dams and reservoirs, nuclear and energy stations, transport infrastructure and so forth) rely on Internet networks, which evidently imposes the necessity of implementing secure and trustworthy communications. Finally, the Internet is also used to perform bank transactions, consume multimedia services and share personal information in the instance of home networks.

Efficiently monitoring and administrating Internet networks assure that the communications are not under risk of sabotage by malicious users or of disruptions due to service collapses. In this regard, network administrators daily cope with vast amounts of traffic flowing through network devices corresponding to thousands of different services, online applications and users. As network topologies tend to be more and more complex and fast events unexpectedly happen at any instant, Network Management is a non-trivial task comprising diverse mechanisms including Traffic Analysis and Active/Passive Network Probing, amongst others [2]. The development of tools that automatically diagnose the status of network devices has been assisting network operators' work for last decades; however, the dynamicity of the Internet imposes arising challenges that need to be addressed. Some of the reasons that make the Internet a challenging scenario for management are:

- The characteristics of topologies and supported traffic vary depending on network environments, locations and even dates [3]. For example, a traffic classifier must be robust to packet losses and multipath effect when it is deployed in a node in the middle of a backbone network, unlike simple home networks. And, evidently, the Internet traffic supported in a home environment considerably differs from the consumed in an enterprise. Additionally, the amount of traffic generated in enterprises during regular dates is not the same as in holiday seasons.
- The emergence of novel services, applications and encryption techniques also imposes obstacles for network managers. Network resource utilization varies depending on the kind of services and applications consumed by users, and demanding applications are regularly launched to the market providing new products and facilities, such as online TV. This fact forces the innovation of Network Management systems to fulfil the emerging demands.
- The Internet is a dynamic worldwide communication infrastructure that is daily increasing the number of interconnected devices. The architecture of the Internet is substantially complex, comprising different transport technologies

(such as, optical and submarine links) that interconnect continents to serve millions of users [4].

- Emerging networking paradigms envision intelligent architectures enabling autonomous Network Management and automatic resource allocation. In the case of the paradigms of Cognitive Networks and Self-Organizing Networks, environment perception is defined as a crucial component for these cutting-edge paradigms [1].
- Finally, the convergence of different communication technologies, such satellite and mobile networks [5], with the Internet is drawing a context with diverse demands to satisfy Quality-of-Service (QoS) of the different technologies. This fact is also the case of the Internet of Things (IoT), which is envisioned as one of the most trending paradigms for remote sensing in the future intelligent cities and Smart Cities.

Network Traffic Classification (NTC) is a key information source to monitor the network activity and/or early detect threats that could jeopardize both regular and critical communications [6]–[9]. This dissertation is focused on NTC based on Machine Learning (ML) techniques, which are attaching a relevant research interest due to their promising outcomes producing efficient classification models. Through this section, we pretend to introduce relevant concepts and bring the readers into the two main research areas that are linked in this thesis: (1) Network Traffic Classification and (2) Machine Learning. Therefore, *Section 1.1* briefly introduces NTC along with other important concepts and discusses its relevance for Network Management. Later, NTC based on ML and its essentials are introduced in *Section 1.2*, meanwhile general methodological aspects in ML are described in *Section 1.3*. The research motivation and objectives pursued in this dissertation are presented in *Section 1.4*. The research methodology followed to identify the research questions is presented in *Section 1.5* and, finally, *Section 1.6* presents the organization of this dissertation.

1.1. Traffic Classification for Network Management

NTC aims at associating Internet flows with the applications or protocols that generate them. It constitutes an important source of information for Network Management and dynamic resource allocation [10], since many events can be efficiently detected via monitoring the traffic supported in network nodes (i.e. Denial-of-Services attacks or Internet link disruptions). Furthermore, NTC can assist in other relevant networking tasks and mechanisms that are quite interesting for Internet Service Providers (ISPs), such as: service arbitration, traffic prioritization, infrastructure planning and so on.

The basic classification objects in NTC are Internet connection flows, which have been defined in different ways during years of Network Management development. Internet connection flows essentially represent information exchanges between tuples origin-destination, typically a client and a server. In this work, we consider Internet flows as the IP packets that transport the messages of a communication in both directions, that is client to server and vice versa. Thus, the packets sharing the same four tuples <*Source IP, Source Port, Destination IP and Destination Port*> during a fixed temporal window are considered belonging to the same connection

flow. As we consider bidirectional connections in this research, the source and destination fields are exchangeable depending on packet directions.

The earliest traffic classification techniques were based on the communication port numbers contained in the transport layer of the OSI model. Under the assumption that numerous applications and protocols use standard and predefined port numbers (e.g. HTTP in port 80, SSH in 22, ftp in 21 and so on), Port-based approaches identify flows via inspecting port numbers and associating them with applications according to the well-known ports defined by the Internet Assignment Number Authority (IANA) [11]. However, the port numbers are parameters that can be configured by users, constituting an important handicap for Port-based Classification. Thus, certain users and applications can easily evade this control mechanism masking Internet connections beyond port numbers used by other well-known services. In fact, port evasion is very often performed by Peer-to-Peer (P2P) applications to avoid being detected. Additionally, P2P applications are quite resource-consuming, since their communication scheme is employed by very demanding services, such as: video streaming and data sharing applications. These facts caused that Port-based classification became obsolete during last decades.

In order to provide a more accurate classification, Deep Packet Inspection (DPI) approach arose as one of the most prominent solutions [8]. DPI tools examine the data contained into IP application layers seeking fixed binary patterns. This approach assumes that some information patterns are signature for certain applications (i.e. the string "HTTP GET" for HTTP queries), so that application data is inspected and compared to a database containing prefixed application signatures. These approaches have produced outstanding improvements in accuracy respecting Port-based tools; but, conversely, several constraints appeared for DPI. Firstly, the use of encryption techniques; as the pointed data are scrambled, pattern seeking turns out challenging for encrypted connections. Secondly, high-speed networks normally support millions of connections per second with vast amounts of packets at single nodes, so that inspecting all packets for all connection flows is very complex and computationally prohibitive. Although novel DPI approaches are increasing notably their capacities in terms of encryption robustness and computational efficiency, there exists other important issue to overcome yet. In this regard, the most limiting obstacle for DPI is privacy violation, since application layers contain personal, sensitive information about users that is lawful protected [12].

The former NTC methods are not the only approaches proposed in the recent years, alternative traffic classification techniques are: Decoding Protocols [7] and Traffic Identification based on Statistical Patterns [13]. In the first case, certain protocols are assumed to implement their communications according to repetitive message exchange patterns between clients and servers; therefore, they can be detected via observing these patterns. In the instance of NTC based on Statistical Patterns, several indicators (such as: number of packets, time of life or bytes transferred) are observed and statistically modeled, and inferring methods are used to identify incoming unknown connections. Next section introduces NTC based on ML and discusses important methodological considerations about ML that underpin some of the advanced methodological techniques applied in this dissertation.

1.2. Network traffic classification based on Machine Learning

One of the most remarkable pioneers in ML research was Arthur Samuel in 1959 [14], who defined ML as: *"the field of study that gives computers the ability to learn with being programmed in an explicit manner"*. The fast-paced development of intelligent techniques and algorithms providing computers with the capacity of learning from experience has opened numerous lines of application to solve problems in diverse research areas [15].

Nowadays, ML provides a wide, varied number of tools that enable processing information at several levels to achieve the final goal of modeling a system, phenomena or problem. These techniques cover from data preprocessing methods to learning algorithms, which can learn from the internal knowledge contained in training data samples. In short, given a system that reacts with a response to certain inputs, ML aims at modeling the internal system behavior via supplying learning algorithms with representative examples about the problem.

There exist two main learning paradigms according to the nature of the addressed problem and the availability of the response to predict: supervised and unsupervised learning. Whether the response is known and is measurable during data acquisition, the problem can be solved from a supervised perspective, in which human intervention is needed to create a ground truth. On the contrary, unsupervised learning assists in solving problems in which the response is unknown and/or unmeasurable. Finally, some problems can be approached from a semi-supervised perspective, which combine assumptions from the two former paradigms.

Regarding the nature of the response to predict, three main problems can be solved using ML: (1) Regression, (2) Classification problems and (3) Clustering/Segmentation. In the instance of regression problems, the response to model is continuous and can take infinite values (numerical measures, such as prices, temperature and so on); meanwhile, the response takes categorical and discrete values in classification problems (i.e. categorizing objects in an image, identification of types of genes, etc.). Finally, clustering refers to problems in which the aim is to identify groups of samples according to their similarities (this is the case of marketing segmentation applications). Although NTC can be also solved from a semi-supervised perspective, we assume it as a supervised classification problem with the objective of associating connection flows to their generating applications, whose ground truth is established via alternative accurate means.

Regarding NTC based on ML, the first researchers that experimented with ML on network traffic were A. McGregor et al. and A.W. Moore and D. Zuev. The former presented in April of 2004 an approach based on the Expectation Maximization clustering algorithm to categorize Internet connections [16]. The objective of this approach is not to classify flows according to Internet applications, but cluster them regarding their network properties (packet size, duration, inter-arrival time and so on). A few months later, A.W. Moore and D. Zuev presented a supervised approach to classify complete TCP connections according to their applications in [17]. The authors trained the Naïve Bayes algorithm using a dataset with 248 features. This

dataset has become very popular, being used by many other authors to experiment with [18]–[20].

In 2006, L. Bernaille et al. presented the concept of Early NTC in [21], consisting in classifying connections flows using the minimal number of packets as possible. The authors employed the distant-based clustering algorithm K-Means to identify Internet applications using the few first packets at beginning of TCP connections. This research work was afterwards extended in [22] testing more clustering algorithms and including features based on inter-arrival times, jitter and packet sizes. The classification models presented in these works exhibited promising accuracies greater than 90%.

Then, N. Williams et al. experimented with different supervised algorithms [23], including: Bayesian Net, C4.5 Decision Tree, Naïve Bayes and Naïve Bayes Tree. The dataset employed in this work was composed by 22 predictors and the models were trained after applying Correlation-based Feature Selection (FS). In [24], J. Erman et al. compared two clustering algorithms (K-Means and DBSCAN) to other previous approaches. Later, T. Auld et al. [25] proposed an approach based on Bayesian Neural Networks and compared it to the Naïve Bayes and multi-layer perceptron classification algorithms. Although substantial advances in ML-based NTC must be performed to achieve feasible classification models, the earliest research shown that ML constitutes a promising solution for accurate, efficient and privacy-respectful NTC.

In this section we have reviewed the most pioneering research that constitutes the advent of this field, and a more extensive revision of the most recent works is provided in subsequent sections of this dissertation. Below, we present the general and most relevant methodological aspects when a problem is approached from an ML perspective.

1.3. Methodology in ML

When a classification problem is formulated from an ML perspective, unavoidable methodological steps must be considered to successfully solve it. Figure 1 presents the most essential methods and materials in ML, from the problem statement until the final predictive model is built. In Figure 1, different colors are employed to differentiate between methodological processes and the materials they produce.

The first and one of the most crucial steps is *Problem Definition*. Through *Problem Definition*, both inputs and outputs (*Predictors & Responses*) of classification objects (instances or samples) must be identified and clearly defined. The predictors and responses contain the basic knowledge about classification objects, which will be processed by ML algorithms to generate the sought predictive models. This step also defines how to collect and process the *Metadata* of the system or problem to model. In this dissertation, each classification object represents a bidirectional connection flow consisting in a tuple $<x, y>$, including the statistical attributes used as inputs and the responses, which are the protocols and applications that generate the connection flows.

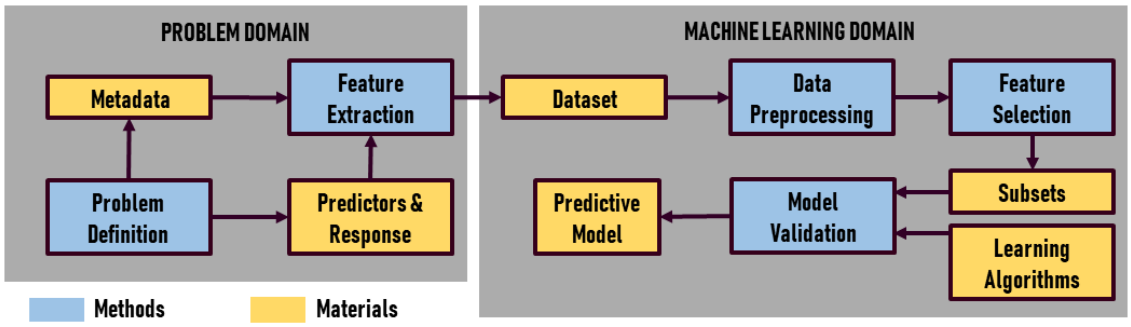

Figure 1. General Methods & Materials in ML

Feature Extraction takes as inputs the collected *Metadata* from the problem, which is processed in order to adequate the information to be computerized according to the guidelines defined in *Problem Definition*. A critical phase in *Feature Extraction* is ground-truth creation, which consists of assigning the actual response (classes or labels) to each instance. *Feature Extraction* must be efficient and accurate, since issues in this phase could alter the representation of classification objects leading to inaccurate models. Furthermore, data collection and processing may be very time-consuming depending on the approached problem. As a result of *Feature Extraction*, a *Dataset* ready to be fed to ML techniques and *Learning Algorithms* is yielded.

In some cases, *Data Preprocessing* is needed to adapt the classification objects to suitable format to be processed by *Learning Algorithms*. This process can include encoding literal features to numerical formats, transforming or processing predictors to generate new attributes (*Feature Engineering*), data normalization and resampling techniques, amongst others.

Feature Selection (FS) is a relevant step in ML, which enables more efficient and accurate predictive models. When *Metadata* collection is performed, it is very common to include features or attributes that could result irrelevant or redundant for the modeling task. Optimizing the inputs to learning algorithms normally results in much better models, since redundant and irrelevant attributes detriment the learning process inputting noise and leading to overfitting on certain classes [26]. Thereby, *FS* produces *Subsets* from the original dataset. Another important advantage of *FS* is that reducing the number of predictors speeds up training and classification processes, turning out faster and more interpretable models.

Through *Model Validation*, the different *Subsets* selected in *FS* are assessed according to a validation strategy and to one or more performance metrics [27]. The validation approach defines how the dataset is used to train and evaluate the performance of classifiers. The simplest validation approach is randomly selecting two datasets, one used for training and the other to assess predictive model performances. However, we have employed more sophisticated validation methods so as to adapt our experiments to arising methodological considerations in ML [28] (such as: Class Imbalance or Concept Drift).

Note that the workflow depicted in Figure 1 is not strict, the phases presented can be altered and some phases could be performed in different orders. Additionally, solving a problem using ML necessarily requires a two-fold knowledge on ML and the specific domain of the addressed problem. Through *Section 4 "Thesis*

Methodology", we describe in detail the methods applied during this dissertation. Having a preliminary view of the general methodology in ML, we present the research motivation and objectives pursued in this thesis below.

1.4. Research Motivation & Objectives

The last advances in ML have propelled the application of these algorithms in order to solve complex problems. ML is being constantly developed via identifying emerging methodology concerns, proposing novel algorithms or spreading out its application domains**.** Focusing on classification problems, one of the most challenging research lines is how to improve model performances when classes are not equally distributed in datasets. This phenomenon is known as Class Imbalance, and it is also an intrinsic feature in NTC. The appearing of novel and complex learning algorithms, such as ensemble and cost-sensitive algorithms, has opened research opportunities in different problems.

Regarding NTC, a crucial open issue is defining consistent and efficient classifiers to achieve real-time traffic identification, as the Internet imposes demanding requirements in terms of accuracy, but also of latency and classification speed. High-speed networks normally support very high transmission rates, what requires optimal classification models. Also, enhancements in FS and Extraction algorithms are a hot research line in ML-based NTC, since providing optimal subsets conduct to more efficient traffic classifiers with a stronger robustness against Class Imbalance.

Finally, never seen before technological advances in communication system envision a technology convergence that will have a significant impact on the observed traffic. That is the case of paradigms as the Internet of Things (IoT), that connects thousands of sensors through regular Internet networks. Thus, proposed network classifiers have to be tested in modern and demanding network environments and data.

Accordingly, we define the following research objectives and questions to be solved in this dissertation:

Regarding model creation for NTC, defining a consistent set of attributes providing the enough knowledge to obtain accurate and fast classification models is crucial. Considering the former, we formulate the following questions:

Q1. What attributes are the most informative and relevant for early NTC? How can it efficiently perform Feature Selection on NTC dataset to achieve effective models?

Focusing on learning algorithms, ensemble techniques have exhibited better performances compared to single estimators for many classification problems, however they were not exhaustively analyzed for ML based NTC. Accordingly, several insightful research lines arise:

Q2. Can ML approaches accomplish similar classification performances to previous NTC approaches? Can ensemble algorithms efficiently perform early NTC according to the accuracy and latency requirements in high-speed networks?

Finally, considering that traffic distributions are notably different between Internet applications, the following questions about Class Imbalance in NTC can be formulated:

Q3. How does Class Imbalance affect Internet traffic classifier performances? How does Class Imbalance vary amongst different network environments? Is it possible to solve Class Imbalance in NTC using advanced ML techniques (such as: data-level resampling, advanced ensemble structures and cost-sensitive algorithms)?

All the present research questions are addressed and answered through this dissertation. As major outcomes of the experiments performed, three scientific manuscripts have been published in top-tier journals in the field and extra contributions interesting for the research community have produced in form of software, analysis and materials (Collection of Attributes, NTC datasets, etc.). All the contributions resulting from this research are thoroughly presented and discussed in subsequent sections and the research articles produced.

1.5. Research Methodology

Regarding the research methodology, we briefly present in Figure 2 the main steps applied to identify *Research Questions* and contribute in solving them.

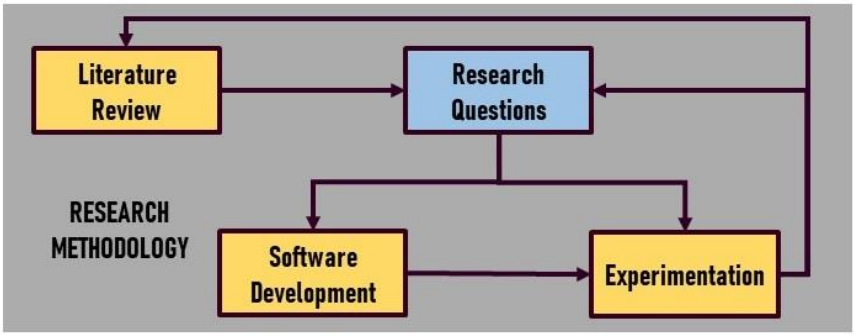

Figure 2. Research Methodology

Literature Review. Through this phase, we have reviewed the latest and most relevant literature in ML-based NTC in order to identify open research lines and gather methodological considerations. This process has required a two-folded knowledge acquisition, reviewing literature specific for both ML and NTC. As a result, the *Research Questions* to solve during through this dissertation are formulated.

Software Development. During this phase, the necessary software was designed and developed with the aim of solving the *Research Questions* formulated. This software derived from coping with the different methodological steps in ML (see Section 1.3), from *Problem Definition* to classification *Model Validation*.

Experimentation. This phase consists of experimenting with state-of-art algorithms and our own algorithm proposals. During the performed experiments, we have assumed different methodologies comprising from dataset creation to several model validation schemes. The resulting observations leaded to worthy asset

observations to solve the *Research Questions*. The whole process was iteratively performed in order to contribute to different problems in this research area.

After presenting the general research workflow followed, we present the structure of this dissertation.

1.6. Thesis Organization

The present dissertation is structured as follows: *Section 2 "Thesis Framework & Contributions"* provides an overall view on the contributions achieved and relates them to the ML workflow; *Section 3 "State of the Art"* reviews the latest research work in ML-based NTC; *Section 4 "Thesis Methodology"* describes the different methodological aspects applied in this research, and the general conclusions and future research are presented in *Section 5 & 6*, respectively. Finally, the research manuscripts composing the publication compendium are annexed at the end of this dissertation.

2. Thesis Framework & Contributions

The modality of this dissertation is the compendium of publications in JCR-indexed journals in the field of Computer Science. Through this research, three manuscripts have been published in top-tier journals, and additional asset were produced in form of different contributions that could be interesting for the research community.

After presenting the ML methodology and the research questions, the contributions of this dissertation are presented and related to the different ML steps in Figure 2. Accordingly, there exist two types of contributions (Figure 3): (1) *Major Contributions* that are highlighted with thicker squares, and (2) *Minor Contributions* represented by narrower squares. *Major Contributions* compose the main asset produced in this research, meanwhile *Minor Contributions* are worthy original materials shared with the research community.

Since this original research addresses ML-based NTC from *Problem Definition* to *Learning Algorithms*, the derived contributions are closely related to the ML workflow and comprise different elements: Scientific Manuscripts, Generated Knowledge, Software and NTC datasets. Below, we describe in detail all *Minor* and *Major Contributions* and the connections between them.

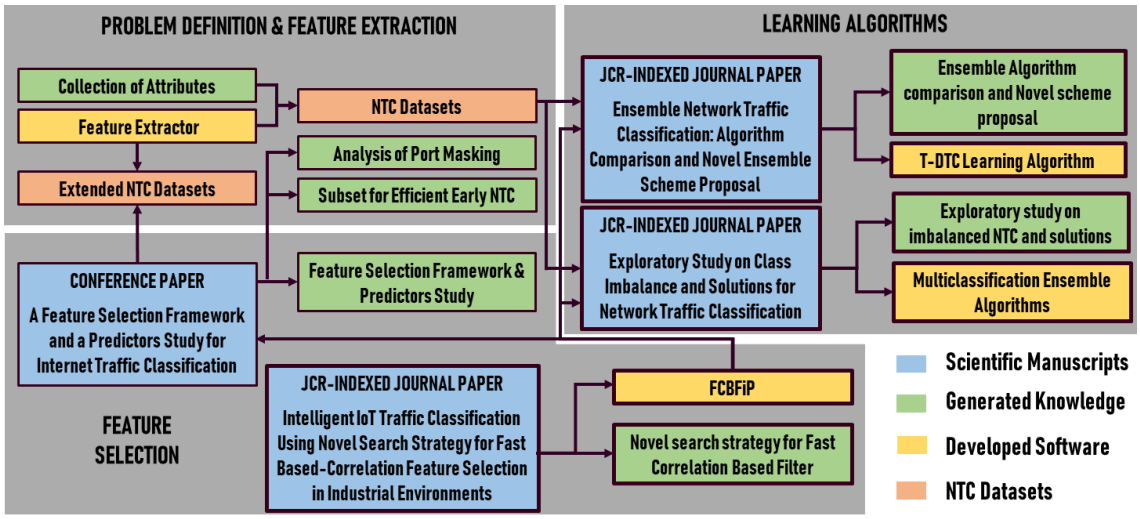

Figure 3. Minor and Major Contributions of this dissertation

2.1. Minor contributions

The *Minor Contributions* compose the generated materials that are interesting for the research community in ML-based NTC, but without constituting the most relevant output presented in the compendium of articles.

When a ML classification problem is approached, the first crucial step is consistently defining the input predictors and the output response (*Section 1.3*). As result of an exhaustive literature revision, potential predictors from IP headers were identified and compiled in a *Collection of Attributes*. The provided *Collection of Attributes* was

used and extended in subsequent research stages. Finally, the classification granularity was also defined in this phase to conform the final classification objects.

After defining the classification objects for our problem, a *Feature Extractor* was implemented to create early *NTC Datasets* from traffic data contained in PCAP network traces. As the studied network environments presented different privacy requirements, several versions of this software were developed. Due to the scarcity of available *NTC datasets* for scientific experimentation, our *NTC Datasets* are shared with other researchers via emailing the authors to facilitate model comparison and innovation. Finally, an extra version of the *Feature Extractor* was developed to generate time-series NTC datasets that were used in a third-party research [29].

In addition to the three scientific manuscripts that compose the publication compendium, a conference paper [30] was presented and published in the First International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2019). Through this article titled *"A Feature Selection Framework and a Predictors Study for Internet Traffic Classification"*, we observe different kind of predictors for early NTC and provide a *FS Framework*. Furthermore, we discuss in this manuscript which are the best attributes for this classification task and relate their performances to network features. To this end, the preliminary *NTC Datasets* were extended to include more predictors, including raw attributes, statistics and Fast Fourier Transform (FFT) components. As result of the application of our FS method to the *Extended NTC Datasets*, a reduced *Subset of Attributes for Efficient Early NTC* is provided. Finally, an *Analysis of Port Masking* is carried out to assess this detrimental effect when port numbers are included as predictors in the classification model.

The presented *Minor Contributions* constituted essential and worthwhile materials to carry out the experimentation that produced the *Major Contributions* presented below. The most of those materials are made available to the research community through emailing the authors or in the GitHub repository [31].

2.2. Major contributions

The *Major Contributions* yielded through this thesis have been materialized in three scientific manuscripts published in top-tier Computer Science journals.

In the first paper titled *"Intelligent IoT Traffic Classification Using Novel Search Strategy for Fast Based-Correlation Feature Selection in Industrial Environments"* [32], the *FS* stage is approached via proposing a novel search scheme as modification of the well-known Fast Correlation Based Filter algorithm [33] (FCBF). The novel FS algorithm, called FCBF in Pieces (*FCBFiP*), was tested in different classification datasets from diverse research areas (including Anomaly Detection in Internet networks) and its applicability to emerging Internet Traffic Classification was discussed. The results show that the proposed search strategy significantly speeds up the selection process preserving similar performances to the previous versions of FCBF [33]. Our FS filter is afterwards employed in subsequent experimentation stages to generate optimal models for ML-based NTC. We make the implementation of *FCBFiP* available in [34].

As second *Major Contribution* presented in the paper *"Ensemble Network Traffic Classification: Algorithm Comparison and Novel Ensemble Scheme Proposal"* [35], we provide an original comparative analysis of well-known ensemble algorithms for early NTC and propose a new ensemble scheme called Tailored Decision Tree Chain (*T-DTC*). During this research, more than nine learning algorithms are compared in terms of accuracy and latency for different network environments after applying *FCBFiP*, showing that *T-DTC* exhibits the best performances for early NTC. The implementation of the proposed ensemble scheme is available in [36]. Furthermore, the *Collection of Attributes* is presented and described in this manuscript.

The third *Major Contribution* comprises an exhaustive study on Class Imbalance and solutions for our *NTC Datasets*, and it is presented in the article *"Exploratory Study on Class Imbalance and Solutions for Network Traffic Classification"* [37]. Through this *Exploratory Study on Imbalanced NTC and Solutions*, we discuss and characterize the Class Imbalance issue relating it to network conditions and performances for ML-based traffic classifiers. Several approaches to deal with imbalanced class distributions are compared for different NTC datasets, including: 21 data-level techniques, advanced ensemble algorithms and cost-sensitive solutions. Furthermore, this is the first time that some data-level algorithms are mixed with Boosting training schemes to compensate Class Imbalance in multiclassification problems to the best of our knowledge. Consequently, the proposed combination of Tomek Link and Boosting yielded the best results for imbalanced early NTC. Finally, this manuscript describes methodological considerations that must be considered in Class Imbalance contexts, and that have been ignored by the ML-based NTC community up to now. As additional output of this experimental stage, we provide strategies to extend binary Class Imbalance solutions to multiclassification and the learning algorithm implementations in [38].

As summary of all contributions achieved in this dissertation, the following section relates all of them to each other and to the research questions previously formulated.

2.3. Paper Rationale and Research Questions

This section is focused on the relation between the research articles produced in this thesis and the Research Questions formulated in *Section 1.4* (Figure 4).

The first paper written is [32] and it is directly related to *Q1*. This paper was published in IEEE Internet of Things Journal addressing the problem of FS for multiclassification problems. And the application of the proposed FS algorithm to the Internet of Things traffic classification is discussed.

Later, [35] was published in the Journal Elsevier Computer Networks. This manuscript answers *Q1* and *Q2* through observing and comparing several ensemble algorithms assessing both model performances and latency. Additionally, a novel ensemble scheme is proposed to preserve a proper ratio between accuracy and classification time.

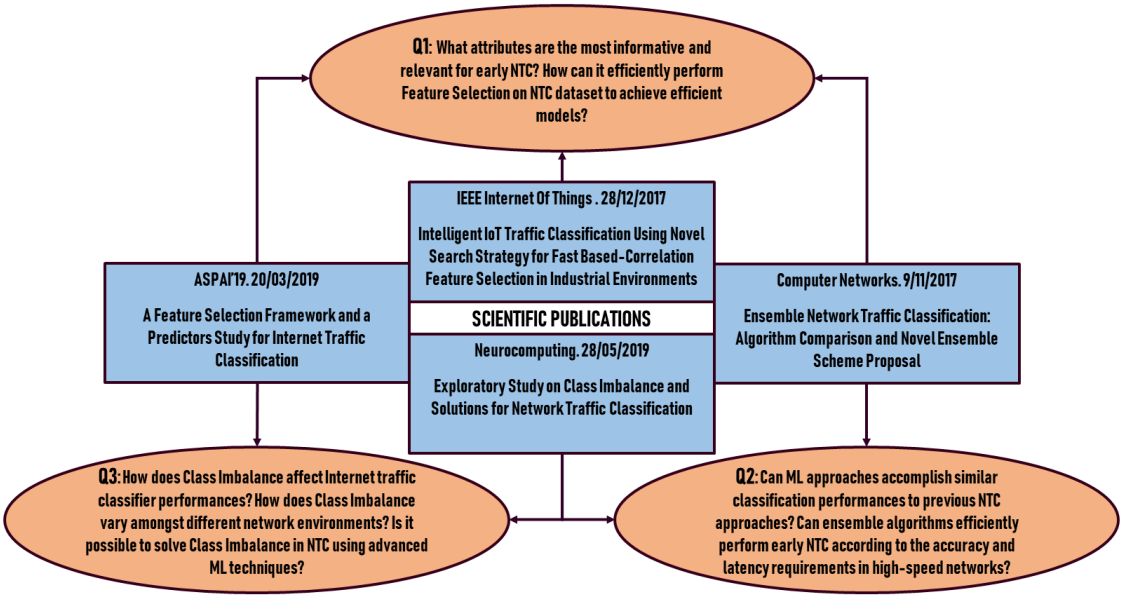

Figure 4. Scientific Manuscripts and their Connections to Research Questions

In the instance of [37], it was accepted in the Special Issue *"Learning in the presence of class imbalance and concept drift"* [39] published in the Journal Elsevier Neurocomputing. This paper copes with *Q2* and *Q3* assessing numerous solutions to Class Imbalance in NTC, including advanced ensemble algorithms.

Finally, [30] addresses the questions *Q1* and *Q3* providing a *Feature Selection Framework* and discussing the capacities of different IP header parameters for early imbalanced NTC. This manuscript was accepted as conference article in ASPAI'19.

All the manuscripts presented in this section are annexed in subsequent sections of this dissertation. Overall, more than 30 predictive models based on the CART Decision Tree algorithm were generated for NTC for two quite different networks scenarios in numerous experiments. The ML techniques considered include preprocessing steps (such as: 21 resampling techniques and FFT attribute generation), 15 ensemble algorithms, one cost-sensitive technique, three model validation approaches and six performance metrics according to several methodological factors from Computer Networks and ML perspective. In the following section, we go in depth into the latest and most advanced literature about ML-based NTC.

3. State of the Art

In *Section 1.2*, we reviewed the most pioneering works in ML-based NTC from which the essentials of this research field derive. These essentials underpin the most advanced research conducted during the recent years, and that is revisited through this section. As each research manuscript composing the compendium summarizes previous research works, we focus here on the literature that we consider the most relevant.

In [19], a comparison of different NTC approaches was conducted. The methods under comparison included Port-based classification, a DPI tool and two ML algorithms: C4.5 Decision Tree and Naïve Bayes. In this case, UDP and TCP flow identification was separately studied focusing on classifiers stability for traffic collected in different dates and locations. The results reported show that C4.5 Decision Tree was the best-performing algorithm preserving an excellent ratio between classification performances and latency.

A. Este et al. approached TCP flow identification using Support Vector Machines (SVMs) in [40]. The presented algorithm exhibited accuracies of 95% for fine-grained classification of specific Internet applications. As extension of their work, A. Este et al. [41] studied packet-level attributes for TCP application classification based on ML. The authors experimented with two supervised (Naïve Bayes and Multilayer Perceptron) and two semi-supervised (K-Nearest Neighbors and Gaussian Mixture Model) learning algorithms probing the predictive power of packet-size related predictors. Other SVM-based classification scheme was presented in [42] for real-time traffic identification. In this instance, R. Yuan et al. studied different kernel functions using a collection of 19 attributes. Radial Basis Function kernel exhibited the best performances; however, the author did not assume the early-NTC predictors.

In [43], several experiments were conducted with the objective of comparing different ML approaches for NTC. The algorithms compared were Bayesian Neural Networks, Decision Trees and Multilayer Perceptron using NetFlow-based features. According to the reported results, Decision Tree was anew found as the most promising approach. Other learning algorithms comparison was performed by Collado et al. [3] including six learning algorithms and various different network environments. The authors demonstrated that network conditions may considerably affect ML-based performances. Additionally, the authors proposed four combination strategies to ensemble different predictive models and boost classification performances.

The effect of packet sampling on traffic classifiers was analyzed in [44] when NetFlow predictors are assumed. The authors employed C4.5 Decision Trees as learning model and extracted ten attributes under different sampling rates proving that packet sampling is detrimental for ML-based NTC. Finally, a new approach to compensate the performance losses was presented exhibiting the best robustness against packet sampling.

T.T.T. Nguyen et al. [45] proposed a technique based on C4.5 Decision Tree and Naïve Bayes for continuous interactive flow classification (namely, VoIP and online

game traffic). A sliding window of 25 packets was used to compute statistical attributes at different points of flow connections enabling tracking the interactive connections. Additionally, two preprocessing techniques were proposed for augmenting model performances.

While on FS for imbalanced NTC datasets, a wrapper FS algorithm, called WSU-AUC, was presented in [20]. The authors used Weighted Symmetrical Uncertainty (WSU) information metric to preselect the predictors and AUC-ROC metric along with a base learning algorithm to evaluate subset performances. This research was focused on TCP connections, thus discarding UDP protocols.

One of the first works assessing standard solutions to Class Imbalance in ML-based NTC is [46]. Two data-level and one cost-sensitive technique were compared using Decision Tree as base estimator. Sixteen datasets collected from three different network locations were analyzed considering only TCP connections. The results show different advantages depending on the technique employed, for example Random UnderSampling was found the most beneficial in terms of training time savings; meanwhile, MetaCOST obtained the best results when training data size is large enough.

In [47], the authors analyzed FS filters using data traffic collected from the Internet of China. The authors trained separate classification models for TCP and UDP and proposed a FS algorithm using information gain and information gain ratio.

J. Zang et al. presented the concept of Bag-of-Flows (BoF) in [48], which exploits correlated flow predictions to boost traffic classifiers. The authors trained Naïve Bayes classifiers using full unidirectional flows and posterior probabilities were aggregated to identify connection flows. The new approach was compared to state-of-the-art ML algorithms exhibiting promising results. Later, this research was extended in [49], in which BoF was used to improve 1-Nearest Neighbor performances.

A new classification approach based on server-client interactions was presented in [50]. The main difference respecting others is that the predictors extracted describe information exchange patterns between both sides of connections. Accordingly, six ML algorithm were analyzed using these attributes, proving that Decision Tree was the best-performing algorithm.

In [51], the FS problem for TCP traffic classification was addressed via proposing an algorithm called Local Optimization Approach (LOA). LOA was conceived as result of experimenting with six FS techniques and assessing their stability and goodness using Naïve Bayes as base estimator. Then, A. Fahad et al. presented the Global Optimization Approach (GOA) in their posterior research [52], which combines several FS filters to preselect predictors and wrapper techniques to get the final subset.

With the aim of identifying unknown flows, Zhang et al. [53] proposed a new learning approach that exploits the concepts of Bag of Words and Latent Semantic Analysis. The applications are clustered using k-Nearest Neighbor algorithm, and resulting clusters are merged using statistical and payload-based information. This approach exhibited very prominent performances using protected, sensitive user information, though.

In [54], J. Camacho et al. presented a traffic classifier that uses flow pairing for stable flow and P2P traffic classification. Distant-based algorithms were studied focusing on statistical validation and generalization ability for dynamic traffic. Finally, the authors presented a similarity function, which classifies connections employing IP addresses, port numbers and timestamps.

In [55], D. Li et al. proposed a NTC approach able to rank predictors and provide classification models exploiting the concept of Multi-Task Learning (MTL). The authors compared their proposal to other MTL techniques and ML algorithms for binary classification.

A self-learning approach classification, called SeLeCT, was presented in [56] to model emerging Internet applications using semi-supervised learning. In this case, K-Means algorithm was upgraded with the ability of interactively creating new clusters to represent new traffic classes. The algorithm was tested for TCP long-live connections yielding superior performances than basic K-Means algorithm.

In [57], L. Peng et al. studied the optimal number of packets to perform early TCP flow classification. The datasets were constructed only using packet lengths extracted from IP headers after TCP handshake phase, and several predictive models based on different supervised models were examined. After statistically validate their results, the authors concluded that 5-7 packets are enough for effective TCP flow classification.

Other FS technique for imbalanced NTC was presented in [58]. The algorithm, called Class-Oriented FS, reduces training datasets in two phases. Firstly, the algorithm searches the optimal features for each individual class, and the final subset is selected according to WSU information metric. Additionally, an ensemble scheme was proposed consisting in different models for each class and a weighted classification strategy.

In [59], Valentín Carela et al. presented a classification system composed by three different classification mechanisms, including a Decision Tree model trained with NetFlow predictors. In order to assure the classifier stability on time, an Automatic Retraining System is proposed showing that the proposed system is able to preserve classification accuracy with time.

The identification of zero-day applications was also addressed in [60]. The authors proposed a semi-supervised technique based on K-Means, Random Forest and BoFs to detect new traffic applications. This system has numerous tuning parameters; thus, the authors also presented a procedure for selecting the input parameters.

D. M. Divakaran et al. [61] proposed an intelligent classifier called Self Learning Intelligent Classifier (SLIC), able to retrain itself via exploiting the concept of BoFs. SLIC rebuilds its classification model using an internal dataset that is grown with samples that are consistently identified. The proposed approach was compared to other techniques producing the best outcomes.

A low complexity traffic classifier based on fuzzy nets was proposed in [62]. The classification model uses a 3-layer neuro-fuzzy network and was compared to other state-of-the-art ML algorithms. The reported results prove that the proposed classifier can accomplish similar performances to other algorithms while reducing

considerably the computational complexity. Finally, the authors presented an implementation for hardware platforms.

In [63], a new classification approach based on Markov Models using sequence of messages sizes was proposed for NTC. Naïve Bayes classifiers were used to model each state in application Markov chains and a GMM estimates posterior probabilities to decide the final class. The authors employed Expectation Maximization Clustering to reduce the number of application classes, and the proposal was tested for TCP and UDP connections.

The temporal behaviors of Internet connections were examined in [64], including a clustering analysis. Unlike previous reviewed manuscripts, the aim of this research is analyzing temporal signatures in data traffic. The presented analysis was performed employing flow-level statistics as inputs, fuzzy Gustafson-Kessel method to detect flow patterns and K-Means to classify the different traffic patterns.

A self-adaptive online classifier was presented in [65], the proposed algorithm uses K-Means as estimator and a cluster refinement technique to remove meaningfulness application clusters. The model was retrained using its own predictions, which are selected for retraining according to an inter-cluster conflicts criterion. The algorithm was also tested for network intrusion detection datasets.

Feature Extraction and Selection for optimal and robust traffic classification was approached in [66]. The authors presented a method to extract predictors based on Wavelet Multifractal transformation, and an FS algorithm, called PCABFS, was also proposed. PCABFS essentially uses Principal Component Analysis (PCA) to filter usefulness features according to variance ratios for the resulting components. The authors compared their proposal to other approaches reporting quite positive outcomes for TCP and UDP traffic.

In order to reduce the number of packets processed for classification, a new approach based on expanding vectors was developed in [67]. Traffic Classifier based on Expanding Vectors (TCEV) examines relationships amongst connections regarding the four tuples <*IPsrc, PORTsrc, IPdst, PORTdst*> to detect three levels of relation. Then, expanding vectors are computed for a time windows employing the relation between connections, and several well-known ML algorithms were trained using these expanding vectors.

A novel classification algorithm, called Imbalanced Data Gravitation Based Classifier (IDGB), was studied in [68] for traffic classification under Class Imbalance conditions. This technique uses weights to deal with imbalanced class distributions strengthening minority class detection. IDGB model was compared to classic ML algorithms and solutions to Class Imbalance for binary traffic classification (such as cost-sensitive and data-level techniques).

The identification of Internet video applications was addressed in [69]. Y. Dong et al. presented a two-phase classification approach, which firstly distinguishes symmetrical and unsymmetrical connections and then identifies video applications based on K-Nearest Neighbors models. The authors shown that their proposal outperforms other similar approaches for video application detection.

An ensemble algorithm using Probabilistic Neural Networks (PNN) was presented in [70] pursuing efficiency and flexibility for traffic classifiers. One PNN is trained to identify one type of traffic class, and heuristics rules are applied for final classification according to posterior probability estimates. The authors compared their proposal to other ML approaches showing that multiclassification PNNs are more efficient and robust against dataset size variations than the rest.

The concept of Extreme-Learning (EL) is exploited in the classification approach presented in [71]. The Kernel Extreme Learning Machine (KELM) with a wavelet transform kernel was employed to train the traffic classification model, and a Genetic Algorithm (GA) was also proposed to automatically tune the kernel parameters. The proposed approach produced positive results when it was trained with a balanced version of the dataset used in [72].

Several Deep Learning approaches based on Recurrent and Convolutional Neural Networks were tested for time-series NTC in [29]. The datasets used were extracted from some data traffic employed in this dissertation using 20 packets at the beginning of each connection. The authors conducted several experiments assessing different neural network settings, dataset sizes and lengths for the time-series dataset.

In [73], Decision Trees and a fuzzy multicriteria technique were combined for NTC and Network Anomaly Detection in order to complement the strengths and weakness of both techniques. The new approach, named PROAFTN, extracts fuzzy decision patterns from a Decision Tree model to create the final classification model.

Other FS wrapper method to select optimal subsets for imbalanced NTC was proposed in [74]. The presented technique preselects the most informative predictors using Weighted Mutual Information (WMI) metric, and then the final subset is built training a learning model and evaluating the AUC-ROC performance metric. The presented FS algorithm was validated using 11 well-known learning algorithms and, additionally, a Robust Selection method was proposed to obtain robust and stable subsets.

In [75], an Efficient Feature Optimization Approach (EFOA) is proposed consisting in Feature Generation and Selection methods. The Feature Generation is based on Deep Learning using Deep Belief Networks to generate new predictors from a preselected subset. The synthetized features are afterwards reduced using WSU. EFOA approach was compared to previous FS approaches yielding the best scores using several learning algorithms as base estimator on two well-known data traffic.

A semi-supervised approach with the capacity of self-training was presented in [76]. The proposed classification approach exploits the concept of multi-view ML via creating different sample representations using three component projections techniques: Isomap, Random Projection and Kernel-based PCA. The generated components are used to train different clustering models based on K-Means. The decided clusters are mapped to actual Internet applications using advanced agreement functions. A mechanism using SVMs was employed to refine decision boundaries and interactively retrain the classification models. The presented algorithm was compared to other semi-supervised approaches for Network Intrusion Detection and NTC datasets.

From literature review, several research gaps were found according to the research methodology presented in *Section 1.5*.

Although it is well-known that ensemble algorithms can significantly boost model performances, we noted that there is not a uniform comparison amongst these techniques providing clear observations on their capacities. As aforementioned, high-speed networks are demanding scenarios support high transmission rates, thereby assessing advance learning algorithms considering both accuracy and latency requirements may be very worthy.

The Class Imbalance problem is a hot research topic in general ML and in ML-based NTC. Numerous techniques have been proposed to compensate the detrimental effect of imbalanced traffic class distributions. But, although several authors coped with imbalanced NTC, a consistent analysis of Class Imbalance in NTC datasets and existing solutions is necessary. Also, assuming the most advanced research methods is mandatory to precisely assess the considered techniques.

Additionally, due to the relevance of Feature Extraction and Selection for ML problems, we also consider an important asset contributing to these two processes. That is the reason why we approached the research experiments addressing all phases in the ML workflow, from *Problem Definition* until providing the *Predictive Models* (see *Section 1.3*). Regarding the former, we designed our own collection of attributes and proposed an FS algorithm to reduce it, producing quite positive results as the results presented in our manuscripts reveal.

Finally, we detected that performed research tend to be skewed due to some experimental concerns. In this regard, many researchers focused their research only on certain applications or transport layer, typically TCP, altering dataset compositions. In our research, we have considered the importance of detecting the most diverse range of applications, and not to notably modify the composition of our datasets. In subsequent sections, we present all the methodological aspects assumed in our experiments.

4. Thesis Methodology

Through this section, we present the most important methodological aspects assumed during the different experimental stages of this dissertation. As the scientific manuscripts that compose this thesis present descriptive sections dealing with the methodologies assumed, this section focuses on the most essential and general aspects.

As aforementioned, several essential steps must be followed when an ML classification problem is addressed. The contributions of this thesis comprise diverse artifacts, including software and NTC datasets that are also presented here. However, the most relevant contributions are presented and discussed in the three research articles conforming the compendium of publications. Consequently, the methods and materials assumed in our experiments have been refined according to the feedbacks provided by expert reviewers during the publication processes. Here, we describe common methodological steps and present the derived contributions.

4.1 Network Environments

After defining the response and predictors for the traffic classification model, the metadata extracted from the pointed network environments must be processed to build the training datasets. In our case, the metadata are network traces that contain the activity of a network device or a host at packet level; thus, the collected traffic data are PCAP files containing IP packets.

It is well-known that Internet traffic can notably differ amongst network environments and validating ML classifiers using data extracted from different dates and locations is highly recommended. Therefore, we have considered two quite network scenarios with dissimilar network conditions for experimentation.

Host environment. In this instance, the traffic traces were shared by the CBA research group of UPC BarcelonaTech. The network activity was manually simulated and collected in host computers located in research labs at University of Catalunya. Three network traces were studied corresponding with different computers and collected during periods of time between February and April of 2008. These traffic data were previously employed in the NTC research [44], [77], [78]**.** These datasets are denoted as "HOST" in the rest of this dissertation and our research articles.

Internet Service Provider (ISP) environment. An ISP has cooperated with this research sharing real network traffic collected at their backbone network between January and March of 2017. This ISP provides Internet connection to public and educational institutions reaching millions of users around Spain. Additionally, the node in which the traffic was collected supports transmission rates close to 7 GB/s. So that this environment constitutes a challenging case of study due to the characteristics of the network. Due to privacy and security concerns, the name of the ISP is omitted through this dissertation, and these datasets are denoted as ISP in our papers.

These network environments constitute two quite diverse cases of study in NTC based on ML, enabling the validation of traffic classifiers under different traffic profiles and network conditions. Note that the ISP network traces are very susceptible of suffering packet losses and multipath effect; meanwhile, these conditions are much weaker for HOST traffic. Additionally, the diversity in the Internet applications also differs between environments, since the applications detected in HOST datasets were manually selected in contrast to ISP network traffic that is composed by actual traffic. Further considerations about traffic compositions of the considered network traces are described in detail in our papers.

4.2 Feature Extraction

In order to build our NTC datasets, we have compiled a list of potential predictors for early NTC, and a Feature Extractor software was developed to process the network traces. The Feature Extractor takes as input standard PCAP files and provides a dataset in which each instance is associated to a connection flow. Although the number of attributes was extended in [32], the preliminary set of predictors comprises a collection of 77 statistical attributes along with the application ground truth and it was presented in [35]. According to [21], informative and relevant predictors can be computed processing a few numbers of packets at the beginning of each connection. Consequently, we have computed the collection of statistical attributes using only five packets at the beginning of connections flows to fulfill the early NTC requirement. Although some research focuses only on TCP or UDP connections, we considered both in this dissertation. Figure 5 describes the different modules and processes that compose our Feature Extractor, in which each color corresponds to different Internet connections.

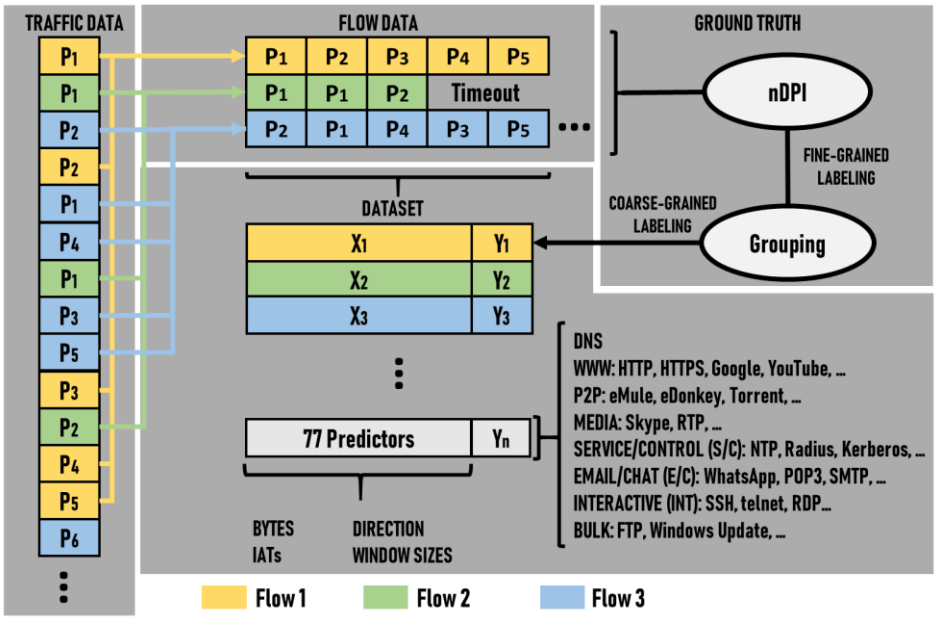

Figure 5. Processes in the Feature Extractor

As Figure 5 shows, the input to the Feature Extractor is packet-level traffic that can suffer packet duplications, out-of-order packets and packet losses. In the first step, the *Traffic Data* is transformed to *Flow Data*, consisting in PCAP files that contain

all packets belonging to a same connection. As aforementioned, we consider bidirectional Internet flows and they are defined for a specific lifetime. Thus, when the lifetime expires without observing new packets for a given connection, the incoming packets with the same source-destination information are considered belonging to a new flow. The flow lifetime was set to 60 seconds for our experiments.

As DPI tools are highly recommended for ground-truth creation due to their precision, *Flow Data* instances are labelled using *nDPI* [79]. We selected *nDPI* because it has probed as one of the most accurate open-source DPI engines [78]. As some of the connection flows were not correctly identified or were identified as "unknown", we performed a second labeling stage based on port numbers.

In the case of ISP datasets, the traffic data was processed in an external server deployed with the aim of preserving users' privacy. Thus, the preliminary software features were extended to include trace anonymization and remote processing. While on HOST datasets, we processed the network traces in a regular personal computer. Finally, the software was upgraded to create the time-series NTC datasets used in [29] and to expand the initial collection of attributes [30]. The datasets employed in this research have been made available for researchers, which is a worthy resource considering the scarcity of NTC datasets.

4.3 Extra Datasets Used in this Research

Although this research copes with NTC, the FS technique *FCBFiP* was tested for extra datasets. Table 1 describes the additional datasets employed in [32].

Table 1. Extra Datasets used in this dissertation

	Description
Orange Churn Prediction [80]	Orange dataset is a collection of information about costumer churn in a telecom company.
KDD99 [81]	KDD99 is a well-known dataset in network anomaly detection. The dataset contains instances representing normal and attack connections.
LSVT Voice [82]	LSVT Voice is a dataset employed in Parkinson diagnosis via voice signals. The dataset records voice signal indicator and the evolution of the patient to predict.
CNAE-9 [83]	CNAE-9 dataset contains text information about the activity of Brazilian companies with the objective of categorizing their activity.

4.4 Feature Selection Techniques

During this research, numerous FS methods were considered for ML-based NTC. Although one of our contributions is a FS algorithm (*FCFBiP*), we have also employed additional methods including wrapper and filter algorithms. FS filters assume information-based metrics to evaluate the importance of each attribute, meanwhile wrapper methods score predictors employing classification performance metrics and learning algorithms. The description of *FCBFiP* is detailed in [32], and it was applied in the other articles conforming the compendium. Whilst the rest of algorithms were integrated in the *FS Framework* presented in [30]. Table 2 contains all FS techniques employed in this dissertation.

Table 2. Feature Selection Algorithms employed during this research

	Description
FCBF [84]	FCBF uses the correlation metric Symmetrical Uncertainty to assess the relevance and redundancy in datasets.
FCBF# [33]	FCBF# modifies the search strategy of FCBF to achieve a more accurate attribute selection.
FCBFiP [32]	FCBFiP is a modification of FCBF whose objective is speeding up the selection process preserving accuracy performances of FCBF.
MRMR [85]	MRMR selects features according to maximum relevancy and minimum redundancy criterion.
CIFE [86]	CIFE maximizes the joint class-relevant information by reducing the class-redundancies.
CNIM [87]	CNIM assumes Conditional mutual information metric to select the features that maximize the class mutual information.
ICAP [88]	ICAP evaluates interaction information between attributes and labels for fast context-dependent attribute selection
MIFS [89]	MIFS performs a greedy selection of predictors by assessing the mutual information between classes and other features.
DISR [90]	A new information metric, Double Information Symmetrical Relevance, is used to evaluate attributes for classification.
JMI [91]	JMI uses Joint Mutual Information in order to reduce the data space.
MIN [92]	MIN individually evaluates the importance of attributes using Mutual Information Maximization approach.
Wrappers [32]	We employed several performance metrics and learning algorithms to assess them when they are input as unique predictor

4.5 Learning Algorithms

The objective of this section is briefly introducing the learning algorithms analyzed in all experiments conforming this thesis. Amongst all these techniques, there are basic learning algorithms, ensemble techniques, data-level methods and advanced algorithms to deal with Class Imbalance.

Base classifier. The Decision Tree algorithm has shown as one of the most suitable solutions for early NTC due to its excellent ratio between accuracy and latency [6], [18], [23]. Although other learning algorithms (such as: SVMs or Logistic Regression) were eventually employed in this research; we chose the CART Decision Tree as base learning algorithm in the most of our experiments. Decision Tree model classification problems via splitting the data space and evaluating a specific information-based metric in order to create tree-shaped hierarchical rules describing data distributions. The version of the algorithm used in our experiments employed GINI index as objective function, whose formula for c classes is Equation 1 and where p_i denotes classes probabilities. Note that GINI index tends to zero when a data region is populated by only one class, meanwhile it will be one when class diversity of the samples is higher.

$$GI = 1 - \sum_{i=1}^{c} (p_i)^2 \qquad (1)$$

Ensemble Algorithms. Ensemble Algorithms are learning techniques composed by various base classifiers that interact to create complex models according to advanced training and classification strategies. These algorithms have exhibited excellent accuracy improvements compared to base estimators in numerous classification

problems. However, using ensembles leads to slower training and classification times, which could be detrimental for early classification in high-speed Internet networks. In [35], we analyze well-known ensemble algorithms focusing on classification performances and latency and proposed a novel ensemble scheme (T-DTC) for early NTC. Table 3 summarizes the algorithms employed for our experiments.

Table 3. Ensemble Algorithms employed during this research

	Description
OneVsRest [93]	One classifier is trained per each class and the predictions are performed according to the maximum class posterior probability.
OneVsOne [93]	One classifier is trained for each pair of classes and the final class is assigned according to a majority voting strategy.
Error-Correcting Output-code [94]	Binary codes are associated to each class and one classifier is trained for each bit; finally, new samples are projected to the binary space and the closest label is assigned.
Adaptive Boosting [95]	A set of classifiers are sequentially trained associating misclassification costs for training samples. Then, new samples are classified according to weighted majority.
Bagging Algorithm [95]	A set of classifiers are trained using different training sets randomly sampled from the original dataset and labels are assigned by majority voting.
Random Forest [96]	A set of trees are trained using different sets with different attributes and instances sampled from the original dataset. Finally, classes are assigned by majority voting.
Extremely Randomized Trees [97]	It is a version of Random Forest in which the splits generated trees during training are completely random, instead of selecting the most discriminative thresholds.
Tailored Decision Tree Chain [35]	Base classifiers are ordered and trained to distinguish only one class, such that each classifier acts as sample filter for its successor.

Class Imbalance techniques. Class Imbalance in ML is a hot issue consisting in model performance degradation due to imbalanced class distributions in training datasets. The state-of-the-art learning algorithms assume that classes are equally represented in datasets, leading to bias when this assumption is not fulfilled. Performance deteriorations normally affect more severely the classes that have a low representation in datasets, meanwhile majority classes are well-modeled. Due to the nature of the Internet, service and application classes are highly imbalanced in NTC datasets. This problem is addressed in the manuscript [37], in which a wide number of solutions to Class Imbalance were analyzed. In the presented research, we studied data-level algorithms, one cost-sensitive technique and advanced ensemble algorithms combining boosting and resampling techniques. To the best of our knowledge, some of the ensemble structures analyzed in this manuscript are applied to a real-world imbalanced multiclassification problem for the first time. Consequently, we had to develop strategies to adapt the algorithms preliminary designed for binary problems to working with more classes. Table 4 contains the algorithms to cope with Class Imbalance employed in our experiments, a more detailed description is presented in [37].

Table 4. Algorithm to deal with Class Imbalance

Data-Level Oversampling	Random OverSampling (ROS), Synthetic Minority Oversampling TEchnique (SMOTE), ADAptive SYNthetic algorithm (ADASYNC)
Data-Level Undersampling	Random UnderSampling (RUS), Near Miss (NM), Condensed Nearest Neighbor (CNN), Tomek Link (TL), One Sided Selection (OSS), Edited Nearest Neighbor (ENN), Neighborhood Cleaning Rule (NCR), Instance Hardness Threshold (IHT)
Data-Level Hybrid Sampling	SMOTE-TL, SMOTE-ENN
Advanced Ensemble Algorithms	Easy Ensemble (EE), Balance Cascade (BC), ROS+Boosting, SMOTE+Boosting, RUS+Boosting. TL+Boosting
Cost-Sensitive Approach	METAcost

4.6 Model Validation & Performance Metrics

This thesis is composed by various research manuscripts that have been published in different journals and one international conference. During the publication processes, expert reviewers' feedbacks were gathered and considered in fulfilling the methodological requirements. As a result, different model validation approaches were assumed in the different experiments that compose this research. In this regard, *Model Validation* in ML is an essential step consisting in estimating classifier performances for future samples in the problem. The simplest validation approach consists of splitting the original dataset in two subsets, one for training and other for validation. However, different methodological considerations had to be considered to adapt the experiments to Class Imbalance conditions [28], [98]. Table 5 presents the validation approaches used in this research. As part of *Model Validation*, we also applied non-parametric statistical validation methods to assess the validity of our observations when different learning algorithms were compared for ML-based NTC. The statistical validation methods applied are the Friedman's Test and Holm's Post-hoc correction method [99], [100].

Table 5. Model Validation Approaches

	Description
K-Fold Cross Validation	The original dataset is randomly divided in k folds without considering any data information, and each fold is used for testing while rest for training.
K-Fold Stratified Cross Validation	The original dataset is divided in k folds preserving class distributions, and each fold is used for validation in each iteration while rest for training.
Distributed Optimally-Based Stratified CV	The k folds are created preserving class and data distributions to reduce covariance between folds.
Time-separated validation	A dataset collected at certain date is used for training, meanwhile other data traffic collected in the same network point but at different date for validation.

During the research presented here, classification performances were evaluated according to different metrics. Although other metrics (such as F1 score) were eventually employed in some experimental stage, below we introduce the most relevant metrics assumed in this dissertation.

Overall Accuracy (OA). OA metric assesses the classification model performances in terms of samples correctly labeled. Namely, OA is the percentage of samples

correctly classified as Equation 2 expresses, being TP_i the true positives for class i and $\#Samples$ the number of samples in the dataset.

$$OA = \frac{\sum TP_i}{\#Samples} \quad (2)$$

Byte Accuracy (BA). From a Network Management perspective, it is interesting to quantify the amount of information that traffic classifiers detect precisely. Thus, we assumed BA metric in our experiments, which is the percentage of bytes correctly classified as Equation 3 shows.

$$BA = \frac{Bytes\ classified\ correctly}{Total\ bytes\ captured} \quad (3)$$

Geometric Mean (GM). In imbalanced classification, OA may not be a suitable performance metric to evaluate model performances, since high accuracies in majority classes can bind poor performances on minority ones. Therefore, we employed GM on the accuracies computed for each individual class. Equation 4 presents the formula for GM for n classes, in which ACC_i represents the accuracy on each class. In addition to the metrics presented, individual accuracies (ACC_i) were also reported in some of our manuscripts.

$$GM = \sqrt[n]{\prod ACC_i} \quad (4)$$

AUC-ROC. The Receiver Operating Curve (ROC) was conceived with the aim of precisely validating classifiers over Class Imbalance conditions. ROC is a graphical representation of binary classifier specificity when the decision threshold varies. In order to have a numerical indicator of this characteristic, the Area Under Curve (AUC) is assumed as metric to evaluate the quality of ROCs. As this metric is defined for binary problems, we aggregated each class AUC using the mean, but also individual AUC-ROCs were reported in some of our experiments. Figure 6 depicts several examples for different classification performances, where the black curve represents the optimal behavior and the orange the ROC produced by a random classifier.

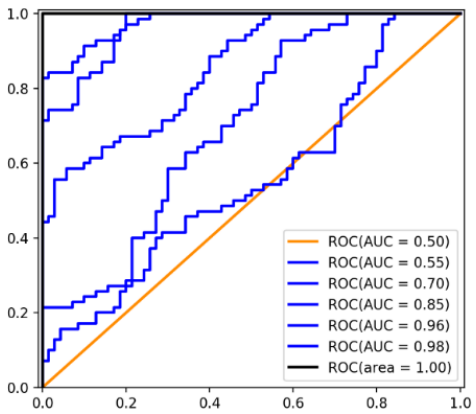

Figure 6. AUC-ROCs for different model performances

4.7 Employed Tools

All the software implementations developed in this dissertation were coded in *Python2.7* programmed language. In order to fulfil the different necessities during the research, several libraries were employed depending on their purposes. Table 6 presents the different libraries used.

Table 6. Software Libraries & Tools employed in this research

	Purpose	Used for
Numpy [101]	Scientific Programming	Scientific Data processing, Feature Extraction and Algorithm Development
Pandas [102]	Statistical Data Analysis	Dataset Processing and Manipulation
Scapy [103]	Network Traces Processing	PCAP Network Traces Processing in Feature Extraction
Scikit-learn [104]	Machine Learning Library	Learning Algorithms and Data Preprocessing
Imbalanced-learn [105]	Imbalanced ML Library	Resampling and Advanced Ensemble Algorithms
Scikit-feature [106]	FS Library	Integrate FS filters in our FS Framework

Table 7. Summary of the methodologies applied in our research articles

[32] Intelligent IoT Traffic Classification Using Novel Search Strategy for Fast Based-Correlation Feature Selection in Industrial Environments

Datasets	Churn Prediction, KDD99, CNAE-9 and LSVT voice
Model Validation	K-fold Cross Validation ($k = 10$ & $k = 5$)
Performance Metrics	AUC-ROCs and F1 Score
Statistical Validation	-
Feature Selection	FCBF, #FCBF and FCBFiP
Learning Algorithms	Support Vector Machines, Logistic Regression, CART Decision Tree

[35] Ensemble Network Traffic Classification: Algorithm Comparison and Novel Ensemble Scheme Proposal

Datasets	ISP and HOST datasets
Model Validation	K-fold Stratified Cross Validation ($k = 10$)
Performance Metrics	OA, BA and $ACC_i s$
Statistical Validation	Friedman's Test and Holm's post-hoc correction method
Feature Selection	FCBFiP
Learning Algorithms	CART Decision Tree and Ensemble Algorithms

[37] Exploratory Study on Class Imbalance and Solutions for Network Traffic Classification

Datasets	ISP and HOST datasets
Model Validation	Distributed Optimally-Based Stratified CV ($k = 5$)
Performance Metrics	OA, BA, GM, $ACC_i s$ and AUC-ROCs
Statistical Validation	Friedman's Test
Feature Selection	FCBFiP
Learning Algorithms	CART Decision Tree and Class Imbalance Techniques

[30] A Feature Selection Framework and a Predictors Study for Internet Traffic Classification

Datasets	ISP datasets
Model Validation	Time-Separated Validation
Performance Metrics	OA, BA and GM
Statistical Validation	-
Feature Selection	FCBF, #FCBF, FCBFiP, MRMR, CIFE, CNIM, ICAP, MIFS, DISR, JMI, MIN
Learning Algorithms	CART Decision Tree and Ensemble Algorithms

4.8 Summary of Methodologies

Through this dissertation numerous ML techniques and methods were analyzed in the different experiments performed. As the main contributions of this dissertation are presented in JCR-indexed journals, the methodologies applied during this research work have been adapted to the feedback provided by expert reviewers in ML and Internet Networks. Table 7 presents a summary of the methods and materials assumed in each research article, including the conference article [30].

5. General Conclusions

In this research, we have addressed the problem of Internet NTC from a ML perspective. Through the numerous experiments reported in the scientific manuscripts composing this compendium, we have analyzed a wide number of advanced learning techniques and proved that ML has promising prospects for achieving efficient traffic classifiers. From a perspective of Computer Networks, ML-based traffic classifiers are interesting alternatives to overcome the limitations of traditional NTC approaches, since they accomplish similar classification performances to DPI tools without privacy concerns. Thus, research efforts in conducting advanced experimentation in ML-based NTC are required, and this thesis aims at contributing in this direction.

This dissertation constitutes a case of study in which traffic data were successfully collected, analyzed, processed and modelled from quite dissimilar network environments. One of the network environments examined is a backbone in an ISP network, which constitutes a real-world challenging problem. The network data employed was collected recently, so that they are composed by the most current Internet applications and protocols. The ground-truth creation was performed using the most accurate open-source DPI tool, and the early NTC requirement was fulfilled in our NTC models.

Regarding the research questions formulated in this dissertation (see *Section 1.4*), we provide below the general observations.

Firstly, we designed a Collection of Attributes as input to our models and, then, we proposed a FS filter to efficiently reduce the problem dimensionality. Our FS proposal was preliminary compared to previous FS based on correlation using classification datasets extracted from different problems, including a Network Intrusion Detection dataset. The conducted experiments show that our proposal outperformed the rest of algorithms and that it is able to reduce the data space preserving, and even increasing, model performances. Later, the proposed FS algorithm was applied to our NTC datasets showing that it provides excellent results for this modeling task and confirming that our Collection of Attributes has substantial predictive power for early NTC. Afterwards, a study on different kinds of predictors was performed employing an important number of FS techniques, and a FS Framework was proposed as a result. Through this analysis, we identified the strengths and weaknesses of different IP-header parameters in modeling different traffic classes and provided a subset producing excellent performances for imbalanced datasets. Finally, we confirmed the risk of suffering performance losses when port numbers are included in the classification models.

Regarding the second question formulated, we have experimented with advanced models based on CART Decision Trees corroborating that ML classifiers can accomplish similar performances to DPI approaches. Additionally, we have observed numerous learning techniques and proposed novel algorithms to overcome the limitations of the state-of-the-art ones. Our first experiments on NTC datasets comprised an analysis of ensemble algorithms for early NTC in high-speed networks. Through these experiments, we analyzed the most popular ensemble algorithms in

terms of model performances and latency, and a novel ensemble algorithm was presented with the aim of preserving the accuracy improvements of ensemble learning without significant latency costs. The proposed algorithm was statistically compared to state-of-the-art techniques proving its excellent performances respecting to the rest.

In order to address our last research question, we performed an exploratory study on Class Imbalance for our NTC datasets. Internet traffic is highly imbalanced, since some applications and services are much more consumed than others depending on network environments. Considering two different networks, we were able to identify and evaluate performance degradations on ML classifiers and link them to network conditions and features. We found that some network scenarios are more susceptible to Class Imbalance than others, and that imbalanced class distributions produce performances losses in all environments to a greater or lesser extent. Additionally, we experimented with advanced learning algorithms to compensate performance losses produced by imbalanced class distributions, including techniques that were never analyzed for real-world multiclass problems. Through the last scientific manuscript, we found various benefits from applying different solutions to Class Imbalance. For example, RUS was found quite useful for reducing the dataset size and speeding experiments. Whilst, the best-performing algorithm in terms of accuracy was the combination of TL and ensemble boosting techniques.

Generally, this thesis shows that ML constitutes a prominent solution to build Internet traffic classifiers. Through the experiments reported here, we approached all the processes necessary to provide accurate early NTC models, from network data collection until classification model creation. The models reported in this research exhibit competitive results comparing to the DPI approach used as baseline, meanwhile performing a privacy-respectful and time-efficient traffic classification. In terms of the applicability to next-generation networks, our findings contribute to achieve efficient and privacy-respectful classifiers in high-demanding environments. Emerging network paradigms rely on network monitoring mechanisms and, therefore, accomplishing better performances in NTC improves the prospects of these paradigms. Also, the effectiveness of the approaches presented here facilitate its integration in hardware platforms, which is other interesting research line. Although this dissertation constitutes an important piece of knowledge in pursuing feasible traffic classifiers based on ML, this research also opens new research questions and opportunities that are discussed in the next section.

6. Future Research Opportunities

Despite the advances achieved during the years of research on ML-based NTC, many experimentation lines are still open. As a result of the thorough literature review performed, we identified interesting research opportunities that are presented and discussed below.

Although several advanced techniques have been presented to get efficient traffic classifiers, the reported results are very often limited to offline experimentation without tests in real-time environments. This experimental deficiency is found in numerous research works, including ours; therefore, it may be very worthwhile to perform online experiments in real Internet networks via embedding the proposed classification models into hardware devices.

The Internet is a very complex environment continuously evolving, since it supports emerging traffic and changes its topology daily. This fact imposes the necessity of self-learning traffic classifiers with the main capacities of detecting emerging traffic and retraining itself to refit classification models when network conditions change. In this regard, zero-day application detection and efficient retraining mechanisms are demanded to expand the capacity of ML-based traffic classifiers.

When a supervised algorithm accomplishes its limits in terms of model performances, a way of upgrading its predictive power is including more informative predictors. Thus, advanced research on finding more effective predictors could be also interesting and worthy for the research field. This research objective may be achieved via Feature Engineering or examining new network-level attributes to represent classification objects. Another research opportunity related to the flow classification objects is the granularity in traffic class representations. In our research we applied a label grouping to reduce the number of classes and this step is commonly performed in other research works. Label grouping has the advantage that learning algorithms model better the generated classes but, conversely, the classification granularity is considerably decreased. Thus, finding ML mechanisms to achieve a finer classification granularity would a great asset.

Furthermore, ML is fast evolving with the appearing of new learning mechanisms that may be also very interesting for upgrading traffic classifiers. That is the case of Multiview learning, in which classification objects are represented by different views [107]. This learning mechanism has shown very successful in diverse research problems, however the research work in NTC is quite scarce. Other novel learning mechanism that could constitute a promising solution for ML-based NTC is MTL. In MTL different close-related modeling tasks are solved using a common model for all classification tasks [108]. This learning mechanisms could be very useful to improve the granularity of traffic classification models based on ML.

Through the literature review performed in this dissertation (see *Section 1.5*), we identified significant methodological differences in the reviewed works that hinder the consistent comparison amongst approaches. For example, some researchers focused their experiments only on TCP connections, and others altered the class distributions. Furthermore, some authors assumed different validation approaches,

which skew the comparison between works. Therefore, standardizing some research methods may facilitate the development of classifiers more and more advanced.

Finally, the convergence of different technologies using the Internet as communication core opens new research stages. Thus, the research advances accomplished through this dissertation could be extended to other problems in Computer Networks, such as Mobile traffic modeling and IoT traffic detection.

List of References

[1] M. A. Khan, S. Peters, D. Sahinel, F. D. Pozo-pardo, and X. Dang, "Understanding autonomic network management : A look into the past , a solution for the future," *Comput. Commun.*, vol. 122, no. May 2017, pp. 93–117, 2018.

[2] D. C. Verma, *Principles of Computer Systems and Network Management*. Boston, MA: Springer US, 2009.

[3] 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.

[4] O. Courtois and C. Bardelay-Guyot, "Architectures and management of submarine networks," in *Undersea Fiber Communication Systems*, Elsevier, 2016, pp. 343–380.

[5] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica, "Mobile Traffic Analysis: a Survey," vol. 18, no. 1, pp. 1–38, 2015.

[6] 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.

[7] A. Callado, A. Callado, C. K. Member, G. Szabó, B. Péter-gerö, and J. Kelner, "A Survey on Internet Traffic Identification . A Survey on Internet Traf fi c Identi fi cation," vol. 11, no. November 2015, pp. 37–52, 2009.

[8] M. Finsterbusch, C. Richter, E. Rocha, J. A. Müller, and K. Hänßgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2014.

[9] 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.

[10] M. Dashevskiy and Z. Luo, "Network Traffic Classification and Demand Prediction," in *Conformal Prediction for Reliable Machine Learning*, Elsevier, 2014, pp. 231–259.

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

[12] R. Bendrath, "Global technology trends and national regulation: Explaining Variation in the Governance of Deep Packet Inspection," in *International Studies Annual Convention*, 2009, vol. 15, no. 18.

[13] K. Yogo, R. Shinkuma, T. Konishi, S. Itaya, and S. Doi, "Coverage area

management for wireless sensor networks," *Int. J. Netw. Manag.*, no. 22, pp. 1–11, 2012.

[14] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, Jul. 1959.

[15] T. M. Mitchell, *Machine learning*. McGraw-Hill, 1997.

[16] A. Mcgregor, M. Hall, P. Lorier, and J. Brunskill, "Flow Clustering Using Machine Learning Techniques," pp. 205–214, 2004.

[17] 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.

[18] 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.

[19] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Networks*, vol. 53, no. 6, pp. 790–809, 2009.

[20] H. Zhang, G. Lu, M. T. Qassrawi, Y. Zhang, and X. Yu, "Feature selection for optimizing traffic classification," *Comput. Commun.*, vol. 35, no. 12, pp. 1457–1471, 2012.

[21] 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.

[22] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," *Proc. 2006 ACM Conex. Conf.*, pp. 6:1--6:12, 2006.

[23] 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.

[24] J. Erman, M. Arlitt, A. Mahanti, I. C. Methodologies, and P. Recognition, "Traffic Classification Using Clustering Algorithms," pp. 281–286.

[25] 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.

[26] I. Guyon, A. Elisseeff, and A. M. De, "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

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

[28] 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.

[29] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for

Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[30] S. E. Gómez *et al.*, "A Feature Selection Framework and a Predictors Study for Internet Traffic Classification," in *First International Conference on Advances in Signal Processing and Artificial Intelligence*, 2019, no. March, pp. 20–22.

[31] S. E. Gómez, "GitHub - SantiagoEG," 2018. [Online]. Available: https://github.com/SantiagoEG?tab=overview&from=2018-02-01&to=2018-02-28. [Accessed: 10-Jul-2019].

[32] 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.

[33] B. Senliol, G. Gulgezen, L. Yu, and Z. Cataltepe, "Fast Correlation Based Filter (FCBF) with a different search strategy," *2008 23rd Int. Symp. Comput. Inf. Sci. Isc. 2008*, 2008.

[34] S. E. Gómez, "GitHub - SantiagoEG/FCBF module," 2018. [Online]. Available: https://github.com/SantiagoEG/FCBF_module. [Accessed: 10-Jul-2019].

[35] 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.

[36] S. E. Gómez, "GitHub - SantiagoEG/TEC module," 2018. [Online]. Available: https://github.com/SantiagoEG/TEC_module. [Accessed: 10-Jul-2019].

[37] S. E. Gómez, L. Hernández-Callejo, B. C. Martínez, and A. J. Sánchez-Esguevillas, "Exploratory study on Class Imbalance and solutions for Network Traffic Classification," *Neurocomputing*, vol. 343, pp. 100–119, May 2019.

[38] S. E. Gómez, "GitHub - SantiagoEG/ImbalancedMulticlass," 2018. [Online]. Available: https://github.com/SantiagoEG/ImbalancedMulticlass/tree/master. [Accessed: 10-Jul-2019].

[39] N. Chawla, "Learning in the presence of class imbalance and concept drift," *Neurocomputing*, vol. 343, pp. 1–2, May 2019.

[40] 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.

[41] A. Este, F. Gringoli, and L. Salgarelli, "On the Stability of the Information Carried by Traffic Flow Features at the Packet Level," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 3, p. 13, 2009.

[42] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate Internet traffic classification," *Inf. Syst. Front.*, vol. 12, no. 2, pp. 149–156, 2010.

[43] 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.

[44] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification," *Comput. Networks*, vol. 55, no. 5, pp. 1083–1099, Apr. 2011.

[45] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and Continuous Machine-Learning-Based Classification for Interactive IP Traffic," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1880–1894, Dec. 2012.

[46] 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.

[47] J. Yang, J. Ma, G. Cheng, Y. Wang, L. Yuan, and C. Dong, "An Empirical Investigation of Filter Attribute Selection Techniques for High-Speed Network Traffic Flow Classification," pp. 541–558, 2012.

[48] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 5–15, 2013.

[49] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 104–117, 2013.

[50] N. F. Huang, G. Y. Jai, H. C. Chao, Y. J. Tzang, and H. Y. Chang, "Application traffic classification at the early stage by characterizing application rounds," *Inf. Sci. (Ny).*, vol. 232, no. 22, pp. 130–142, 2013.

[51] A. Fahad, Z. Tari, I. Khalil, I. Habib, and H. Alnuweiri, "Toward an efficient and scalable feature selection approach for internet traffic classification," *Comput. Networks*, vol. 57, no. 9, pp. 2040–2057, 2013.

[52] A. Fahad, Z. Tari, I. Khalil, A. Almalawi, and A. Y. Zomaya, "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion," *Futur. Gener. Comput. Syst.*, vol. 36, pp. 156–169, 2014.

[53] 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.

[54] 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.

[55] D. Li, G. Hu, Y. Wang, and Z. Pan, "Network traffic classification via non-convex multi-task feature learning," *Neurocomputing*, vol. 152, pp. 322–332, 2015.

[56] L. Grimaudo and M. Mellia, "Self-learning classifier for Internet traffic," *Infocom, 2013 …, 2013*, vol. 11, no. 2, pp. 144–157, 2014.

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

[58] 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.

[59] V. Carela-Español, P. Barlet-Ros, O. Mula-Valls, and J. Solé-Pareta, "An Autonomic Traffic Classification System for Network Operation and Management," *J. Netw. Syst. Manag.*, vol. 23, no. 3, pp. 401–419, Jul. 2015.

[60] 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.

[61] 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.

[62] A. Rizzi, A. Iacovazzi, A. Baiocchi, and S. Colabrese, "A low complexity real-time Internet traffic flows neuro-fuzzy classifier," *Comput. Networks*, vol. 91, pp. 752–771, 2015.

[63] A. Hajjar, J. Khalife, and J. Díaz-Verdejo, "Network traffic application identification based on message size analysis," *J. Netw. Comput. Appl.*, vol. 58, pp. 130–143, 2015.

[64] F. Iglesias and T. Zseby, "Time-activity footprints in IP traffic," vol. 107, pp. 64–75, 2016.

[65] H. R. L. M. N. Marsono, "Online network traffic classification with incremental learning," *Evol. Syst.*, vol. 7, no. 2, pp. 129–143, 2016.

[66] 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.

[67] L. Ding, J. Liu, T. Qin, and H. Li, "Internet traffic classification based on expanding vector of flow," *Comput. Networks*, vol. 129, pp. 178–192, 2017.

[68] 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.

[69] Y. Dong, J. Zhao, and J. Jin, "Novel feature selection and classification of Internet video traffic based on a hierarchical scheme," *Comput. Networks*, vol. 119, pp. 102–111, 2017.

[70] S. Dong and R. Li, "Traffic identification method based on multiple probabilistic neural network model," *Neural Comput. Appl.*, 2017.

[71] F. Ertam and E. Avcı, "A new approach for internet traffic classification: GA-WK-ELM," *Measurement*, vol. 95, pp. 135–142, 2017.

[72] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," *Queen Mary Westf. Coll. Dep. Comput. Sci.*, no. August, 2005.

[73] O. Article, "Hybrid multicriteria fuzzy classification of network traffic patterns , anomalies , and protocols," 2017.

[74] M. Shafiq, X. Yu, A. Kashif, B. Hassan, N. Chaudhry, and D. Wang, "A machine learning approach for feature selection traffic classification using security analysis," *J. Supercomput.*, 2018.

[75] H. Shi, H. Li, D. Zhang, C. Cheng, and X. Cao, "An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification," vol. 132, pp. 81–98, 2018.

[76] A. Fahad, A. Almalawi, Z. Tari, K. Alharthi, F. S. Al Qahtani, and M. Cheriet, "Semtra: A semi-supervised approach to traffic flow labeling with minimal human effort," *Pattern Recognit.*, vol. 91, pp. 1–12, 2019.

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

[78] 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, Jan. 2015.

[79] 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.

[80] R. Niculescu-mizil *et al.*, "Winning the KDD Cup Orange Challenge with Ensemble Selection."

[81] "KDD Cup 1999 Data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed: 29-Nov-2017].

[82] A. Tsanas, M. A. Little, C. Fox, and L. O. Ramig, "Objective Automatic Assessment of Rehabilitative Speech Treatment in Parkinson's Disease," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 1, pp. 181–190, Jan. 2014.

[83] P. M. Ciarelli and E. Oliveira, "Agglomeration and Elimination of Terms for Dimensionality Reduction," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*, 2009, pp. 547–552.

[84] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," *Int. Conf. Mach. Learn.*, pp. 1–8, 2003.

[85] Hanchuan Peng, Fuhui Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[86] D. Lin and X. Tang, "Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion," Springer, Berlin, Heidelberg, 2006, pp. 68–82.

[87] F. Fleuret, "Fast Binary Feature Selection with Conditional Mutual Information," *J. Mach. Learn. Res.*, vol. 5, pp. 1531–1555, 2004.

[88] A. Jakulin, "Machine Learning Based on Attribute Interactions," *Thesis*, pp. 1–252, 2005.

[89] R. Battiti, "Using Mutual Information for Selecting Features in Supervised Neural-Net Learning," *Ieee Trans. Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[90] P. E. Meyer, C. Schretter, and G. Bontempi, "Information-Theoretic Feature Selection in Microarray Data Using Variable Complementarity," *IEEE J. Sel. Top. Signal Process.*, vol. 2, no. 3, pp. 261–274, 2008.

[91] H. H. Yang and J. Moody, "Data Visualization and Feature Selection: New

Algorithms for Nongaussian Data," *Adv. Neural Inf. Process. Syst.*, vol. 12, no. Mi, pp. 687–693, 1999.

[92] D. D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," pp. 212–217, 1992.

[93] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, 2000.

[94] T. G. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Jouranal Artifical Intell. Res.*, vol. 2, pp. 263–286, 1995.

[95] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, no. 1/2, pp. 105–139, 1999.

[96] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[97] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[98] 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.

[99] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci. (Ny).*, vol. 180, no. 10, pp. 2044–2064, 2010.

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

[101] "NumPy — NumPy." [Online]. Available: https://www.numpy.org. [Accessed: 10-Jul-2019].

[102] "Pandas." [Online]. Available: https://pandas.pydata.org/index.html. [Accessed: 10-Jul-2019].

[103] "Scapy." [Online]. Available: http://www.secdev.org/projects/scapy/. [Accessed: 10-Jul-2019].

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

[105] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 1–5, 2016.

[106] J. Li *et al.*, "Feature Selection: A Data Perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, Jan. 2016.

[107] S. Sun, L. Mao, Z. Dong, and L. Wu, *Multiview Machine Learning*. Singapore: Springer Singapore, 2019.

[108] Y. Zhang and Q. Yang, "A Survey on Multi-Task Learning," Jul. 2017.

A.1 Journal Paper. Intelligent IoT Traffic Classification Using Novel Search Strategy for Fast Based-Correlation Feature Selection in Industrial Environments

Table A1. JCR-Indexed Paper Information

Title	Intelligent IoT Traffic Classification Using Novel Search Strategy for Fast Based-Correlation Feature Selection in Industrial Environments
Authors	Santiago Egea Gómez, Albert Rego Mañez, Belén Carro, Antonio Sánchez-Esguevillas and Jaime Lloret
Journal	IEEE Internet of Things Journal (IF: 9.515)
Volume	Volume: 5, Issue: 3, June 2018
Publication Date	28 December 2017
DOI	10.1109/JIOT.2017.2787959

Intelligent IoT Traffic Classification Using Novel Search Strategy for Fast-Based-Correlation Feature Selection in Industrial Environments

Santiago Egea, Albert Rego Mañez, Belén Carro, Antonio Sánchez-Esguevillas, *Senior Member, IEEE*, and Jaime Lloret, *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) can be combined with machine learning in order to provide intelligent applications to the network nodes. Furthermore, IoT expands these advantages and technologies to the industry. In this paper, we propose a modification of one of the most popular algorithms for feature selection, fast-based-correlation feature (FCBF). The key idea is to split the feature space in fragments with the same size. By introducing this division, we can improve the correlation and, therefore, the machine learning applications that are operating on each node. This kind of IoT applications for industry allows us to separate and prioritize the sensor data from the multimedia-related traffic. With this separation, the sensors are able to detect efficiently emergency situations and avoid both material and human damage. The results show the performance of the three FCBF-based algorithms for different problems and different classifiers, confirming the improvements achieved by our approach in terms of model accuracy and execution time.**

***Index Terms*—Correlation-based methods, emergency detection, feature selection, filter methods, industry, Internet of Things (IoT), machine learning, multimedia traffic.**

I. Introduction

INTERNET of Things (IoT) pretends to extend sensoring, computation, and communications to every field and object. One of the most important fields where IoT can be applied is on industry. There are many advantages that industry can obtain from IoT, but also there are many challenges to resolve [1], [2]. However, when these challenges are solved, the ubiquity that industry will obtain from IoT will lead to significant improvements on its procedures. For instance, the increase of hazard and emergency detection may currently save millions of dollars wasted due to the losses produced by those emergencies [3].

One of the techniques that can be applied to IoT is machine learning and artificial intelligence [4]–[6]. Machine learning has become popular in the last decades for many fields, from biology to telecommunications. Machine learning provides predictive models that are able to predict or detect responses to problems employing knowledge previously collected in a dataset. Nowadays the learning algorithms are more powerful, and our computing tools are more sophisticated. Despite of these facts, the industry poses new and more complex problems each day, with higher accuracy requirements. Applying machine learning to IoT introduces new constraints like more energy consumption or computation time. In other words, the complexity of these challenges is increasing constantly. These issues force scientists to pay attention, not only to learning algorithm designing, but also to efficient information processing. The majority of learning algorithms are able to model problems more accurately when the input of the classifier is optimal [7]. Thereby, removing useless features is a much recommended practice, and this task is carried out by feature selection methods.

The effectiveness of feature selection has already been proved in numerous works. In fact, these techniques are considered essential in data preprocessing stages [8]. Feature selection consists of selecting the relevant features from the original dataset and remove the rest that could be potentially irrelevant or/and redundant for the problem [7].

The advantages of performing feature selection are well-known [9]: preventing the model from overfitting the training set, thus increasing the accuracy over the test set; reducing both storage requirement and needed computing resources; improving the interpretability of predictive models, since feature selection mitigates the course of dimensionality; and remaining a suitable tradeoff between number of instances and number of features, as this relationship is crucial for some learning algorithms.

According to the way in which the problem is tackled, feature selection methods are mainly split in three groups [9]–[11]: 1) filter methods; 2) wrappers methods; and

Manuscript received September 18, 2017; revised December 21, 2017; accepted December 24, 2017. This paper was supported in part by the Ministerio de Economía y Competitividad del Gobierno de España and the Fondo de Desarrollo Regional within the project Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software under Grant TIN2014-57991-C3-2-P, in part by the Ministerio de Economía y Competitividad in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento within the Project Distribucion inteligente de servicios multimedia utilizando redes cognitivas adaptativas definidas por software under Grant TIN2014-57991-C3-1-P, and in part by the Ministerio de Educación, Cultura y Deporte, through the Ayudas para contratos predoctorales de Formación del Profesorado Universitario FPU (Convocatoria 2015) under Grant FPU15/06837. *(Corresponding author: Jaime Lloret.)*

S. Egea, B. Carro, and A. Sánchez-Esguevillas are with the Communications Systems and Networks Laboratory, Universidad de Valladolid, 47011 Valladolid, Spain (e-mail: santiago.egea.gomez@gmail.com; belcar@tel.uva.es; antoniojavier.sanchez@uva.es).

A. Rego Mañez and J. Lloret are with the Universitat Politecnica de Valencia, 46022 Valencia, Spain (e-mail: alremae@teleco.upv.es; jlloret@dcom.upv.es).

Digital Object Identifier 10.1109/JIOT.2017.2787959

3) embedded methods. Filter methods use a relevance measurement in order to classify the features as useful or not, according to a threshold [9], [12]. Filter methods are computationally very light and, also, they are scalable and independent of the learning algorithm employed in the problem. However, the subset resulted from filter methods is not the optimal one. Furthermore, a criterion has to be chosen for measuring the feature relevance. Therefore, lots of subgroups are included into this category. The other feature selection technique includes the wrappers methods [9], [10], [12]. Their principles are based on the fact that machine learning algorithms are capable of scoring the features during the training process. Once the predictive model is built, we can get a subset for modeling tasks via observing the learning algorithm structure. These methods are slower; since they need to train a classifier and, additionally, the possible subsets have to be validated by cross validation or other validation technique. Furthermore, wrapper methods have difficulties in terms of scalability and have a high risk of overfitting the training set. But they usually produce more accurate subsets than filter methods for a specific classifier.

The most modern techniques are the embedded methods. These techniques are implemented inside the learning algorithm and their search strategy is guided by the learning process. As embedded methods are optimized for a specific learning algorithm, they are faster than wrappers methods and achieve the best subsets; however, they are fully dependent on the used learning algorithm.

The feature selection methods are formed by six properties or phases [10]: 1) initial state of search; 2) creating successors; 3) search strategy; 4) feature evaluation method [13]; and 5) stop criterion.

This paper is focused on filter methods, and namely, on methods based on correlation measurements. The fast correlation-based filter [14] (FCBF) is the most popular of them. Later, a new strategy approach was introduced in [15], this algorithm is known as FCBF#.

In this paper, we introduce a novel search strategy whose goal is to give a tuning parameter that allows users to control both the algorithm computing time and the intercorrelation among the features contained in the resulting subset. With this proposal, we are able to create an optimal subset of features to classify the traffic propagated through an IoT network implemented in an industrial facility. Therefore, the detection of multimedia traffic is improved thanks to this proper selection of the features and can be separated in a better way from the sensor data, increasing the efficiency of the critical and priority use and management of that kind of data. Therefore, applications using that critical data, such as emergency detection will increase their performance. This algorithm is called FCBF in pieces (FCBFiP).

This paper is organized as follows. First of all, in Section II, a review of the current state of IoT and machine learning in the literature is presented. In Section III, we review the prior algorithms and explain our proposal. Next, in Section IV, we describe the experiments carried out to validate our proposal. In Section V, we show and discuss the results obtained for our algorithm and the prior ones by using four different datasets. Finally, in Section VI, we draw conclusions about the results obtained.

II. Related Work

In this section, some of the works related to IoT for industry and machine learning are discussed.

Wan *et al.* [16] proposed and analyze a new entity for production processes in industry called context-aware cloud robotics (CACR). This new entity does an effective load balancing and provides context-aware services in factories. This CACR improves the material handling. In this paper, the architecture of CACR is shown, analyzed and discussed. The results show that CACR, working with decision-making algorithms, works in a more energy-efficient mode and increases the cost-saving during the material handling.

An advantage related to the use of IoT for industry is the reduction of energy-consumption during the production process. These kind of energy-related issues are discussed in [17], where sustainable development and green technologies are the point for saving energy and reducing emissions.

Related to environment, Mehmood *et al.* [18] proposed an artificial neural network in order to save energy and to make the routing scheme more robust. This neural network, called ELDC, has been designed for industry pollution monitoring and increases the lifetime of the nodes by incorporating the features of group-based protocols. The results show that the lifetime of the nodes is increased over 40% compared against other algorithms.

There are some published works related to pollution monitoring and energy saving. In [19], the increase of pollution and carbon footprint problems are discussed and a solution given in terms of routing protocol is proposed. This routing protocol, called secure and low-energy zone-based routing protocol is designed in order to face two problems: 1) energy consumption and 2) security. Taking some assumptions from the features of wireless sensor networks (WSNs), the base station divides the network into zones and clusters, reducing the number of messages. The results show an increase of around 400% of network lifetime. Moreover, the wasted energy is reduced.

Traffic classification and filtering has been deeply applied in several works and fields. A case study is realized by Gupta and Muttoo [20], where the Internet traffic survellance and network monitoring in India is studied. Under the context of preventing terrorist attacks, India is working toward the development of surveillance systems. One of this kind of systems is NETRA, used by the Indian Government to search suspicious keywords from messages in the network. In [20], NETRA is compared against some other similar systems like Dish Fire, Prism, or Echelon. Their work shows how NETRA works and how it filters the messages and traffic. Authors conclude that it shows only a bit weak in spying the content.

This traffic monitoring and processing has been also applied in IoT environments. Zheng *et al.* [21] introduced a nonintrusive traffic data collection for intelligent transportation systems using WSNs. They placed magnetic sensor nodes on the road to collect data from vehicles and obtain the vehicle flow data.

This data is sent to a control center using ZigBee protocol, where the final vehicle flow data is calculated by using filtering and decision-making algorithms. They provide some experiments in order to demonstrate that the method illustrated is reliable. This process is nonintrusive to the transportation systems.

The traffic monitoring can be used to obtain some flows or patterns like it has been done in the previous reference. However, it can also be used for improving the performance of the network. Avvenuti *et al.* [22] proposed a MAC protocol, an extension from B-MAC+ protocol, which reduces the energy consumption for communication in WSNs. This protocol is adaptive and asynchronous. It adapts depending on the observed traffic load and changes its operational parameters. The duty cycle is either increased or decreased attending to the incoming packet number variation. The protocol is distributed into the nodes of the network. The performance evaluation is done through two different simulated scenarios. The results show that the adaptive B-MAC+ protocols achieves a network lifetime from 1.35 up to 2.8 times longer than the standard B-MAC+ protocol.

Furthermore, the collection and analysis of the data are not only used to reduce the energy consumption with MAC-level protocols or to produce new data, but also are used to create a general view of the state of the network. Tang *et al.* [23] introduced a new congestion-aware routing scheme that is based on the traffic information given from the sensors in a WSN. Congestion is one of the most important problems in data networks and the proposal consists on reducing the network delay by being aware of the produced congestion. Moreover, the throughput is also increased. The routing scheme described achieves its goals by using a geographic routing scheme. Therefore, the relay node is selected attending to the sensor node location and the current congestion of the area. The traffic sent by that local area is analyzed and due to that traffic information the algorithm selects the next hope node in the path. The simulations presented in the work show that the end-to-end packets transmission delay is reduced by 50% and the throughput of the network is doubled.

Filter methods are vital to obtain a good performance when taking decisions. In [14], FCBF is presented. A new upgrade is described in [15]. This last method is called FCBF#. They are explained in detail in the next section.

Concerning to the network traffic classification, correlation-based filters have been employed to this modeling task for several years ago. Williams *et al.* [24] provided a comparison between learning algorithms, but, additionally, they demonstrated that correlation-based filters are suitable for traffic classification.

Many authors have provided solutions to select the most informative attributes to identify network traffic. In [25], a hybrid feature selection algorithm is presented for high-speed networks. The algorithm consists of two selection phases, the less relevant and most redundant attributes are prefiltered using a new metric called weighted symmetrical uncertainty at the first stage, and later, the final subset is provided training different learning algorithms and evaluating the area under curve performance metric. The authors reported significant improvements in terms of true positive rate and false positive rate.

More recently, Fahad *et al.* [26] proposed an novel feature selection scheme to obtain optimal and stable subsets for traffic classification. They discuss the traffic profiling changes, how they affect the classifier performances and propose new metrics to assess the optimality and stability of subsets. In order to avoid performance losses, they present a multicriterion feature selection method called global optimization approach (GOA). GOA combines well-known feature selection techniques to filter out the irrelevant attributes and the resulting subset is processed to extract the stable features based on information theory measures.

The different works commented in this section seeks to improve the performance of the WSN, either by reducing energy consumption or delay or by increasing the throughput and time alive. In order to achieve their goals, the authors proposed new routing schemas, algorithms, or data processing.

In this paper, we work on improving the core of the intelligent network decision. A new filter method based on FCBF is proposed to improve the correlation of the features. Therefore, the algorithms and machine learning tools that make use of it will be able to increase their performance. In other words, it will make the classification and detection algorithms better. The method presented is intended to be used for multimedia traffic classification in IoT for industries. Specifically, in facilities where the data sensed are used for emergency detection and are sent through the network beside the multimedia traffic. The improvement of detection algorithms and special processing of the sensor data will repercute in reducing losses.

III. Fast Correlation-Based Feature Selection

Many researchers have approached the feature selection problem from different viewpoints. Filter methods are underpinned by mathematical and statistical concepts as entropy, mutual information [13] or correlation measurements [27]. Relief algorithm [28] measures the feature relevance, but it is not capable of removing redundant features. Later, correlation-based approaches have been used in order to mitigate features redundancy, like CFS [27]. Afterward, Yu and Liu [14] presented the FCBF algorithm, which speeds up the selection process. FCBF algorithm has been tested in many modeling problems, proving its excellent performance. In [15], the search strategy of FCBF was improved and a stop criterion was included. In our proposal, we implement new capabilities for FCBF. The key idea is to split the feature space in pieces with the same size, compute the redundancy of each feature with a multivariate evaluation method and rank them. Each piece is processed independently. According to the scores assigned to the features and the number of features selected for the resulting subset, the algorithm drops the worst features and includes the rest into the model. The size of the pieces is a design parameter which allows us to control the tradeoff between execution time of the algorithm and intercorrelation of the resulting subset.

A. FCBF Algorithm

FCBF selection [14] uses the symmetrical uncertainty as evaluation method. The symmetrical uncertainty takes some advantages against other correlation measures: is normalized between 0 and 1; detects several kinds of correlations (not only linear correlation); and compensates for information gain's bias.

Symmetrical uncertainty uses the concept of entropy to measure the correlation between features. Given a feature X that can take i different values (x_i) with different occurrences, the entropy of X is defined as

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i)) \tag{1}$$

where $P(x_i)$ is the probability of X to take x_i. The entropy of X given other feature Y is called conditional entropy of X over Y, and is defined as

$$H(X|Y) = -\sum_j P\big(x_j\big)\sum_i P\big(x_i|y_j\big)\log_2\big(P\big(x_i|y_j\big)\big). \tag{2}$$

Now, we define the information gain as

$$\mathrm{IG}(X|Y) = H(X) - H(X|Y). \tag{3}$$

Finally, the symmetrical uncertainty between X and Y is defined as

$$\mathrm{SU}(X, Y) = 2\left[\frac{\mathrm{IG}(X|Y)}{H(X)+H(Y)}\right]. \tag{4}$$

Note that a value $\mathrm{SU}(X, Y) = 1$ indicates a completely correlation between X and Y. Meanwhile $\mathrm{SU}(X, Y) = 0$ indicates that variables are not correlated.

The search strategy used by FCBF sorts the feature space based on the symmetrical uncertainty between each feature and the class. The overall complexity of FCBF is $O(N\log N)$ [14]. And FCBF does not have stop criterion, so that it finishes the search when the whole feature space has been explored. This fact is a shortcoming, since FCBF removes features with no possibility of choosing the number of features desired for the model. Nevertheless, the FCBF efficiency has already been shown [14].

B. FCBF#

FBCF# tries to overcome the above issue, and also modifies the search strategy [15]. A stop criterion has been included in the algorithm by introducing a natural parameter k. When the subset has k features, the algorithm finishes the search and returns the subset. In addition, the search strategy has been changed, so that the process starts removing the irrelevant features during the first iterations. Unlike FCBF, Senliol *et al.* [15] have used a stop counter in their FCBF implementation in order to compare models with same number of features. The results prove that the change in the search strategy improves the model accuracy. However, the algorithm is slightly slower than FCBF.

C. Our Proposal: FCBF in Pieces

Our algorithm, FCBFiP, includes two significant modifications with respect to the previous versions: the feature space is divided in P pieces and the criterion to remove the features is based on a scoring step.

Both FCBF and FCBF# consist of two steps. The first one evaluates the relevance of each feature for predicting the target class, and sorts them in descending order (sequence 1). This step remains in our algorithm and the second one is modified to avoid iterations which go over the whole feature space. At the first step, when two or more correlated features exist, it is expected that they have similar relevance for forecasting the response. Thus, they have to be close in the ordered sequence of features (sequence 1). Then, it is feasible to think that is not necessary to evaluate the redundancy of a variable over the whole feature space but, evaluating the redundancy on its neighboring may be enough. The number of pieces defines the size of the vicinities as

$$V\mathrm{size} = \frac{N}{P} \tag{5}$$

where N is the number of features in the original dataset and P the amount of selected pieces.

In this way, we can save up many operations if P is large. On the other hand, the resulting subset could contain redundant features, as the vicinity size is small. On the contrary, when P is smaller, we will spend more time to process each piece and the resulting subset will present lower intercorrelation among the features included in it. To control the degree of redundancy in the resulting subset may be beneficial depending on the nature of the problem we are modeling. Other advantage of splitting the feature space is that modern programming languages offer tools to parallelize the computation, speeding up the algorithm, since each piece can be processed independently. This fact will be considered for future implementations of FCBiP.

The evaluation method employed for determining the redundancy of each feature is the computation of the mean symmetrical uncertainty (6) between a given feature and its neighbors

$$\overline{\mathrm{SU}}(X_i, V) = \frac{1}{V\mathrm{size}-1}\sum_{j=V;j\neq i}\mathrm{SU}\big(X_i, X_j\big) \tag{6}$$

where V is the vicinity that contains the feature X_i.

In the scoring step, the aim is to classify the features according to its relevance and redundancy into its piece. After computing the mean symmetrical uncertainty for each feature, they are sorted in ascending order (sequence 2). Next, the score assigned to each feature is the sum of the position they occupy in sequences 1 and 2. Finally, FCBFiP removes the features with greater score until the subset contains k features.

The process to obtain the sequence 2 is described in Fig. 1. First, we split the feature space in P fragments. Next, we compute the $\overline{\mathrm{SU}}$ for each feature into its vicinity. Finally, we order the feature space in ascending $\overline{\mathrm{SU}}$ order to get sequence 2.

This approach suffers a crucial limitation. The number of pieces, P, has to be a divisor of N. Thus, when N is

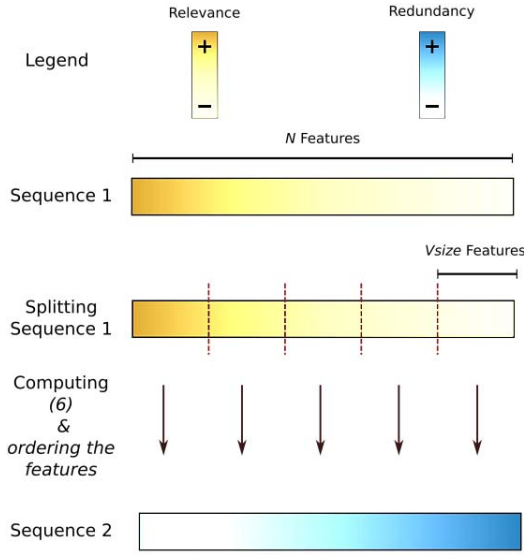

Fig. 1. Description of the process used to obtain the sequence 2.

TABLE I
DATASET INFORMATION

Name	#Features	#Classes	#Instances	Ratio
Small Orange	230	2	50,000	217.39
10% KDD99	41	23	437,000	10658.54
CNAE-9	856	9	1,080	1.26
LSVT voice	309	2	126	0.41

a prime number or has few divisors, a feature preselection using FBCF# is the best solution.

IV. METHODS

In this section, we describe the experiments carried out. We have selected four datasets corresponding with different classification problems and related to areas that could be extrapolated for IoT. Then, we have preprocessed them in order to suit them to the algorithm inputs. These preprocessing steps differ among them, as the formats of the datasets also differ. The following sections go in depth in the experiment setting.

A. Tools

The tools used to perform the experiments were Python libraries. For building the model we used Sklearn [29]. All algorithms were programmed using Numpy [30].

B. Datasets

We chose four datasets. To make the results more general, we looked for datasets whose ratio between #Instances-#Features and origin differ. Also the number of classes to forecast differs. Table I summarizes the characteristics of each dataset.

The Orange [31] dataset was purposed for the KDD cup Orange challenge. Several authors have written about this challenge (e.g., [32] and [33]). This dataset is highly complex, therefore we used a small version of the original dataset. Additionally, we simplified the problem to solve only the churn prediction task, therefore, this problem is a binary classification. In the IoT industry, numerous services are rising up and providers of services will compete in an emerging market. Thus, churn prediction also applies to IoT (as a matter of fact many lines affected if a customer changes the provider).

The KDD99 [34] dataset consists of about 4 370 000 data flows represented by 41 features. And the aim is to identify whether each flow corresponds to a computer attack or to a normal behavior. Other works have already been published using this dataset (e.g., [35]). In this experiment, we used a reduced version of this dataset that includes 10% of all samples (437 000). There are 23 different attacks to predict. Although 41 is prime, this is not a limitation, since the algorithm is capable of detecting this situation and dropping the less relevant feature. IoT traffic goes through network infrastructures to implement the communication between devices. Therefore, the IoT sensors are as sensitive to cyber-attacks as other devices, such as personal computers. Thereby, guaranteeing the security of IoT devices is a must to assure the services. Attack detection via machine learning could be a promising solution for IoT attacks.

The CNAE-9 [36] dataset is extracted from a text mining problem. The dataset contains 1080 free text business descriptions of Brazilian companies [37]. The goal is to classify these descriptions in nine categories. The features are 856 word frequency records. IoT customers are typically enterprises. Therefore, its description is quite useful in order to classify target customers.

The LSVT voice [38] dataset was used to predict Parkinson's disease evolution [39]. It is a binary problem, since persons labeled with "1" are patients whose disease evolution is positive, and "0" the opposite case. This dataset has 309 features corresponding to 126 patients. Thus, the ratio between #Instances-#Features is lesser than 1. Digital home virtual assistants constitute an emerging category of IoT devices. In this context, machine learning models could be employed to monitor patients based on their voice inputs.

C. Preprocessing

Due to the differences between the datasets used in our experiments, we preprocessed each dataset differently.

The small Orange dataset contains artificial variables introduced by the promoters of the challenge. Thus, we have removed the features that only take a value, as they do not give useful information [32]. Also we filled the missing values with the feature mean value in case of the numeric features. This dataset is formed by 40 categorical variables. These variables were encoded with strings to ensure the anonymity of the data. Thus, we have mapped these variables with integer values, including the missing values. Finally, the resulting dataset had 212 variables. To suit the KDD99 dataset to our experiments we randomly shuffled the samples several times, since the instances were sorted by the class to predict. Furthermore, this dataset has three categorical features and the class coded as string. All of them were mapped with integer values. The CNAE-9 dataset also had the instances ordered. Thus the

samples were shuffled randomly in the same way as the former dataset. Besides, the dataset was normalized between 0 and 1, as the classifier used for this dataset is sensitive to feature ranges. The LSVT voice dataset was also shuffled. Additionally, we normalized the dataset between 0 and 1, as the selected classifier requires. Finally, we have carried out a feature selection step, since the number 309 has only two divisors (3 and 103). To get more divisors, we applied the FCBF# algorithm with $k = 306$.

D. Classifers Used

For the Orange dataset, we chose a decision tree classifier because this kind of classifier needs less training time than others. To avoid overfitting the training set, the depth of the decision tree was limited to 6, and the minimum samples per leaf was set to 22.

In the case of the KDD99 dataset, we also used a decision tree with the same parameters as above to decrease the computing requirements for the experiments.

For the CNAE-9 dataset, we modeled the problem by using support vector machines (SVMs). The regularization parameter, C, was fixed to 40.

For the last dataset (LSVT voice), we observed that logistic linear regression slightly outperforms an SVM classifier. Thus, we used logistic regression to tackle this problem. The regularization parameter, C, was set to 1.

For all multiclass problems (KDD99 and CNAE-9), the approach used to assign the final class to the samples was one-vs-the-rest strategy.

E. Model Validation

The measurements to assess the model validity were the F1 score for all problems, except for the Orange dataset. The F1 score was selected due to the fact that it gives information about the model precision and recall [40]. The F1 score is defined as

$$F1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \quad (7)$$

The AUC-ROC score was used for the Orange dataset, because it was the score proposed by the promoters of the challenge [32].

As validation algorithm, we chose k-fold cross validation, since it is a low variance method. The folds were fixed to ten for all datasets; except for KDD99 dataset, we used fivefolds as the dataset contains enough number of samples. All experiments were repeated ten times, and we computed the mean of the resulting scores in order to rank the feature selection algorithms. For the multiclass problems, we computed the mean of the score over all possible classes.

V. Results

Figs. 2–5 present the relevant results obtained from the experiments carried out, both model performance and execution time are shown.

Fig. 2 depicts the results obtained for the Orange dataset. FCBFiP did notably speed up the selection process when the feature space was divided in 106 and 53 pieces. However, they did not get the highest AUC-ROC score, although, in most cases, their performances are quite close to the other candidates. Even, FCBFiP overcame FCBF# when models with 40 and 60 features were chosen. The FCBF algorithm returned a subset with six features. For this subset size, the best results were achieved by FCBFiP with $P = 4$, but the spent time was significantly greater than the other algorithms. Note also that, for a resulting subset with more than 120 features, it was possible to obtain a model with similar performance that FCBF#, but spending much less time. Finally, the global maximum performance was accomplished by FCBFiP with $P = 2$ for a model that included 180 features. However, the time required was much higher than that for the FCBF# algorithm.

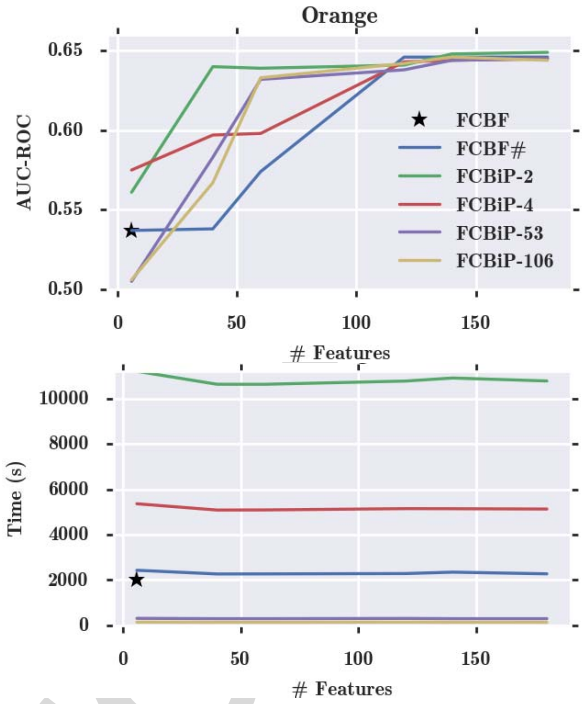

Fig. 2. Performances obtained for Orange dataset.

In the case of the KDD99 dataset, Fig. 3, we note that FCBF# overcame its competitors when ten features are selected in terms of accuracy. However, the FCBiP algorithm obtained better performances than the other ones for models with more than 10 features. FCBFiP with $P = 10$ achieved the highest score for a model with 20 variables and the same happened with FCBFiP with $P = 8$ for 30 features. These results reveal that the intercorrelation among features in a model may be beneficial in specific cases. However, the time spent in these cases was greater than the time spent by FCBF#. The best results in terms of F1 score were obtained using FCBFiP with $P = 5$ for a model with 12 features, but it lasted more time than FCBF#.

Fig. 4 shows the results obtained modeling the CNAE-9 problem. Note that FCBF algorithm yielded a model with 47 features. In this case, the FCBF outperformed the other candidates. The FCBF# and FCBFiP performances increased as the number of features included in the model was gradually

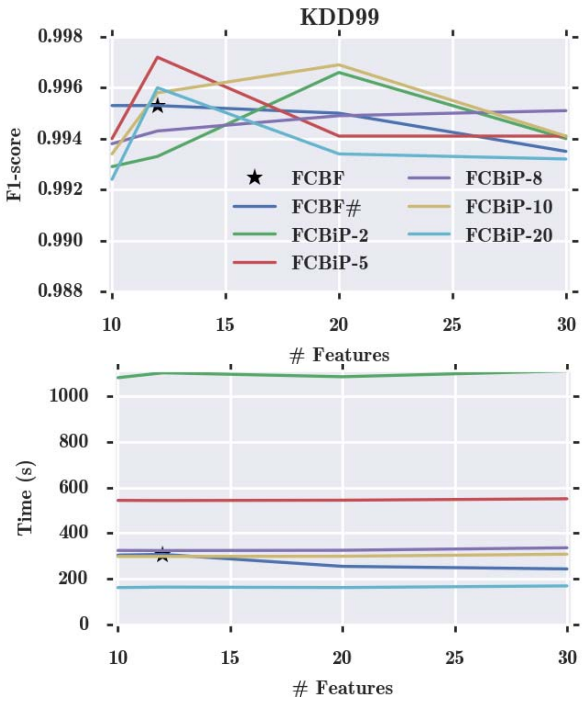

Fig. 3. Performances obtained for KDD99 dataset.

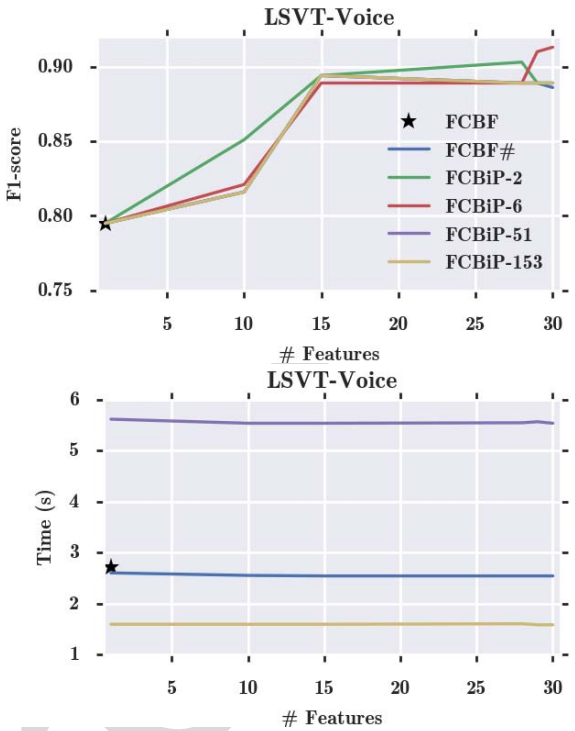

Fig. 5. Performances obtained for LSVT-voice.

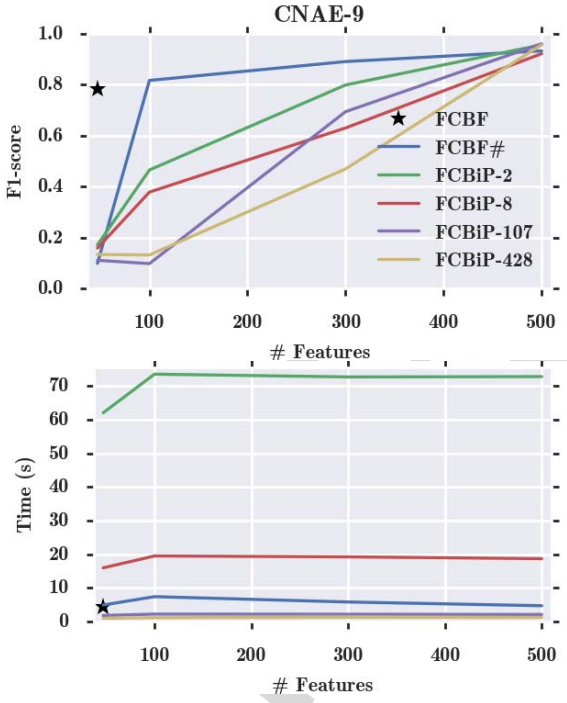

Fig. 4. Performances obtained for CNAE-9 dataset.

raised. Also, the F1 scores obtained when applying FCBFiP differed considerably when the number of pieces varies for models with less than 500 features. In this case, the FCBFiP performances were very poor and were clearly overcome by FCBF and FCBF#. However, the best result was obtained by FCBFiP algorithm with $P = 107$ for a model with 500 features. It achieved higher score than FCBF# but lasting half the time. These results show that penalizing the intercorrelation between features may improve the accuracy of the model for specific cases.

Fig. 5 shows the results obtained by modeling the LSVT voice problem. As FCBFiP-2 and FCBFiP-6 obtained quite high execution times to visualize the plot properly, both temporal curves were excluded from the figure; the execution time was around 145 s for FCBFiP-2 and 51 s in the instance of FCBFiP-6. This dataset presents a ratio between #Instances-#Features lower than 0.5. Note that FCBF returned a subset with one feature. In this case, all algorithms converged in the same solution. That fact may be due to the samples scarcity, since it is related to the available information for the selection and modeling processes. Note that there are more cases in which the different algorithms get the same results, for example when a model with 15 features is selected. For this experiment, the FCBFiP algorithm did not offer great advantages in terms of execution time. Nonetheless, the most accurate model resulted by using FCBFiP with $P = 6$ and 30 features.

VI. Conclusion

In this paper, we review some feature selection filters based on correlation measurements, and we propose a novel approach for providing new functionalities to the FCBF algorithm in order to improve IoT-based intelligent networks in industrial facilities. Our proposal consists of a modification of the original FCBF algorithm, called FCBFiP, by changing the evaluation method of the redundancy and including a scoring process for ranking the variables. The new redundancy evaluation was developed in two steps: first, by splitting the

feature space in P pieces with the same size; and second, by evaluating the feature redundancy in the piece that contains it with a multivariate correlation measurement. This evaluation method allows us to set the number of pieces in which to split the whole feature space, being this parameter able to control both the execution time and the redundancy penalty in the selection process. The scoring process is carried out by ordering the sequences of features according to their relevance and redundancy measurements; assigning the scores according to the position each feature occupies in these sequences; and removing the features that obtain the worst scores. We validated our proposal by comparing our FCBFiP method with the FCBF and FCBF# algorithms.

The datasets selected for the experiments were very different from each other to make the results more generalizable. Additionally, we modeled the problems by using different learning methods for each dataset, namely: decision trees, SVM, and logistic linear regression. The global highest performance for each experiment was achieved by our algorithm in terms of F1 score. Note that best F1 score does not always imply less execution time, parameter for which our algorithm offers a clear advantage. It is possible to obtain a subset with similar performances than the obtained by FCBF or FCBF# but spending much less time. Therefore, we have accomplished a more flexible solution by tuning a new design parameter. Furthermore, we can conclude that a lesser redundancy penalty improved the accuracy of the model built for some of the cases under study. We have found that the ratio between #Instances and #Features actually affects the selection process.

Further work can be done opening new lines for upgrading the FCBFiP algorithm: mixing evaluation methods (e.g., including mutual information scores) and parallelizing operations to speed up the algorithm. Besides, performing more experiments using other datasets might complete and expand the conclusions. For this aim, the code of the algorithm has been published in Gómez [41]. Feedbacks and debug reports are welcome. Moreover, a first implementation can be tested in an IoT environment, using sensor nodes to collect data and FCBFiP algorithm to classify traffic in order to check the increment of performance in the entire IoT system. Nonetheless, this algorithm has already been applied to a network traffic classification task in [42], where we employed FCBF to build consistent subsets in order to identify Internet traffic in two different contexts.

In our future work we will check if this new method of features selection can be used to improve some other typical parameters in IoT networks, like energy consumption of routing decisions.

REFERENCES

[1] M. Garcia, D. Bri, S. Sendra, and J. Lloret, "Practical deployments of wireless sensor networks: A survey," *Int. J. Adv. Netw. Services*, vol. 3, no. 1, pp. 170–185, 2010.

[2] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[3] S. Mannan and F. P. Lees, *Lees' Loss Prevention in the Process Industries: Hazard Identification, Assessment, and Control*. Amsterdam, The Netherlands: Butterworth-Heinemann, 2005.

[4] D. Ventura *et al.*, *ARIIMA: A Real IoT Implementation of a Machine-Learning Architecture for Reducing Energy Consumption*. Cham, Switzerland: Springer, 2014, pp. 444–451. AQ2

[5] R. Xue, L. Wang, and J. Chen, "Using the IOT to construct ubiquitous learning environment," in *Proc. 2nd Int. Conf. Mech. Autom. Control Eng.*, Hohhot, China, 2011, pp. 7878–7880.

[6] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.

[7] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, nos. 1–4, pp. 131–156, Jan. 1997.

[8] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proc. 11th. Int. Mach. Learn.*, pp. 121–129, 1994.

[9] I. Guyon, A. Elisseeff, and A. M. De, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[10] L. C. Molina, L. Belanche, and A. Nebot, "Feature selection algorithms: A survey and experimental evaluation," in *Proc. IEEE Int. Conf. Data Min.*, Maebashi, Japan, 2002, pp. 306–313.

[11] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.

[12] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 245–271, Dec. 1997.

[13] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[14] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proc. Int. Conf. Mach. Learn.*, Washington, DC, USA, 2003, pp. 856–863.

[15] B. Senliol, G. Gulgezen, L. Yu, and Z. Cataltepe, "Fast correlation based filter (FCBF) with a different search strategy," in *Proc. 23rd Int. Symp. Comput. Inf. Sci. (ISC)*, Istanbul, Turkey, 2008, pp. 1–4.

[16] J. Wan *et al.*, "Context-aware cloud robotics for material handling in cognitive industrial Internet of Things," *IEEE Internet Things J.*, to be published. AQ3

[17] G. Han *et al.*, "Recent advances in green industrial networking [guest editorial]," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 14–15, Oct. 2016.

[18] A. Mehmood, Z. Lv, J. Lloret, and M. M. Umar, "ELDC: An artificial neural network based energy-efficient and robust routing scheme for pollution monitoring in WSNs," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[19] A. Mehmood, J. Lloret, and S. Sendra, "A secure and low-energy zone-based wireless sensor networks routing protocol for pollution monitoring," *Wireless Commun. Mobile Comput.*, vol. 16, no. 17, pp. 2869–2883, Dec. 2016.

[20] R. Gupta and S. K. Muttoo, "Internet traffic surveillance & network monitoring in India: Case study of NETRA," *Netw. Protocols Algorithms*, vol. 8, no. 4, p. 1, Jan. 2017.

[21] J. Zheng *et al.*, "Non-intrusive traffic data collection with wireless sensor networks for intelligent transportation systems," *Ad Hoc Sensor Wireless Netw.*, vol. 34, no. 1, pp. 41–57, 2016.

[22] M. Avvenuti, C. Bernardeschi, L. Cassano, and A. Vecchio, "Adapting the duty cycle to traffic load in a preamble sampling MAC for WSNs: Formal specification and performance evaluation," *Ad Hoc Sensor Wireless Netw.*, vol. 31, nos. 1–4, pp. 101–129, 2016.

[23] D. Tang, T. Li, and J. Ren, "Congestion-aware routing scheme based on traffic information in sensor networks," *Ad Hoc Sensor Wireless Netw.*, vol. 35, nos. 3–4, pp. 281–300, 2017.

[24] 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, pp. 5–16, Oct. 2006.

[25] H. Zhang, G. Lu, M. T. Qassrawi, Y. Zhang, and X. Yu, "Feature selection for optimizing traffic classification," *Comput. Commun.*, vol. 35, no. 12, pp. 1457–1471, 2012.

[26] A. Fahad, Z. Tari, I. Khalil, A. Almalawi, and A. Y. Zomaya, "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion," *Future Gener. Comput. Syst.*, vol. 36, pp. 156–169, Jul. 2014.

[27] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 1999.

[28] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Proc. AAAI*, San Jose, CA, USA, 1992, pp. 129–134.

[29] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Feb. 2012.

AQ4 [30] "NumPy—NumPy."

[31] *SIGKDD: KDD Cup 2009: Customer Relationship Prediction*. Accessed: Nov. 29, 2017. [Online]. Available: http://www.kdd.org/kdd-cup/view/kdd-cup-2009

[32] A. Niculescu-Mizil *et al.*, "Winning the KDD cup orange challenge with ensemble selection," in *Proc. Int. Conf. KDD Cup*, Paris, France, 2009, pp. 23–34.

[33] U. Yabas and H. C. Cankaya, "Churn prediction in subscriber management for mobile and wireless communications services," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, USA, 2013, pp. 991–995.

[34] *KDD Cup 1999 Data*. Accessed: Nov. 29, 2017. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[35] M. K. Siddiqui and S. Naahid, "Analysis of KDD cup 99 dataset using clustering based data mining," *Int. J. Database Theory Appl.*, vol. 6, no. 5, pp. 23–34, 2013.

[36] *UCI Machine Learning Repository: CNAE-9 Data Set*. Accessed: Nov. 29, 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/CNAE-9

[37] P. M. Ciarelli and E. Oliveira, "Agglomeration and elimination of terms for dimensionality reduction," in *Proc. 9th Int. Conf. Intell. Syst. Design Appl.*, Pisa, Italy, 2009, pp. 547–552.

[38] A. Tsanas. *Athanasios Tsanas Personal Web*. Accessed: Nov. 29, 2017. [Online]. Available: https://people.maths.ox.ac.uk/tsanas/data.html

[39] A. Tsanas, M. A. Little, C. Fox, and L. O. Ramig, "Objective automatic assessment of rehabilitative speech treatment in parkinson's disease," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 1, pp. 181–190, Jan. 2014.

[40] C. Goutte and E. Gaussier, *A Probabilistic Interpretation of Precision, Recall and F-Score, With Implication for Evaluation*. Heidelberg, Germany: Springer, 2005, pp. 345–359.

[41] S. E. Gómez. (2016). *FCBF_Module*. [Online]. Available: https://github.com/SantiagoEG/FCBF_module

[42] 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. Netw.*, vol. 127, pp. 68–80, Nov. 2017.

Santiago Egea received the Telecommunication Engineering degree from the Polythenique University of Cartagena, Murcia, Spain. He is currently pursuing the Ph.D. degree at the University of Valladolid, Valladolid, Spain.

He is a member of the Communications Systems and Networks Laboratory, as a Technical Researcher involved with national projects in the areas of machine learning and network management. His current research interests include signal processing and machine learning specifically applied to telecommunication networks.

Albert Rego Mañez received the bachelor's degrees in computer science and telecommunications technology engineering and master's degree in telecommunications from the Polytechnic University of Valencia, Valencia, Spain, in 2015 and 2016, respectively. He is currently pursuing the Ph.D. degree at the Polytechnic University of Valencia.

He has authored several papers and participated in some international conferences, both by reviewing papers and being a part of committees. His current research interest includes software-defined networks.

Mr. Rego Mañez was the recipient of an FPU national scholarship.

Belén Carro received the Ph.D. degree in broadband access networks from the University of Valladolid, Valladolid, Spain, in 2001.

She is a Professor with the Department of Signal Theory and Communications and Telematics Engineering, University of Valladolid. She is the Director of the Communications Systems and Networks Laboratory, as a Technical Researcher and Research Manager involved with European and national projects in the areas of service engineering and SOA systems, IP broadband communications, NGN/IMS, VoIP/QoS, and machine learning. She has supervised several Ph.D. students on topics related to personal communications, IMS, and machine learning. She has extensive research publications experience as an Author, a Reviewer, and an Editor.

Antonio Sánchez-Esguevillas (SM'XX) received the Ph.D. degree (Hons.) in QoS for real time multimedia services over IP networks from the University of Valladolid, Valladolid, Spain, in 2004. AQ5

He has been managing innovation with Telefonica (both at the Telefonica I+D-Services line and at Telefonica Corporation), Madrid, Spain. He has also been an Adjunct Professor and a Honorary Collaborator with the University of Valladolid, supervising several Ph.D. students. He has coordinated very large (in excess of 100 million) international research and development projects in the field of personal communication services, particularly related to voice over IP, and Internet Protocol multimedia subsystem. He has over 50 international publications and several patents. His current research interest includes digital services, including machine learning.

Dr. Sánchez-Esguevillas is the Editorial Board member of *IEEE Communications Magazine*.

Jaime Lloret (M'07–SM'10) received the B.Sc. and M.Sc. degrees in physics in 1997, the B.Sc. and M.Sc. degrees in electronic engineering in 2003, and the Ph.D. degree in telecommunication engineering (Dr. Ing.) in 2006.

He is a Cisco Certified Network Professional Instructor. He was a Network Designer and Administrator for several enterprises. He is currently an Associate Professor with the Polytechnic University of Valencia, Valencia, Spain. He is the Chair of the Integrated Management Coastal Research Institute and Head of the active and collaborative techniques and use of technologic resources in the Education Innovation Group. He is the Director of the University Diploma Redes y Comunicaciones de Ordenadores and was the Director of the University Master Digital Post Production from 2012 to 2016. He has authored 22 book chapters and has had over 380 research papers published in national and international conferences, international journals (over 140 with ISI Thomson JCR).

Dr. Lloret was the Internet Technical Committee Chair (IEEE Communications Society and Internet Society) from 2013 to 2015. He has been the Co-Editor of 40 conference proceedings and a Guest Editor of several international books and journals. He is the Editor-in-Chief of *Ad Hoc and Sensor Wireless Networks* (with an ISI Thomson impact factor), the *International Journal of Networks Protocols and Algorithms*, and the *International Journal of Multimedia Communications*. He is an IARIA Journals Board Chair (eight journals) and is (or has been) an Associate Editor of 46 international journals (16 of them with an ISI Thomson impact factor). He has been involved in over 400 Program Committees of international conferences and over 150 Organization and Steering Committees. He leads many national and international projects. He is currently the Chair of the Working Group of the Standard IEEE 1907.1. He has been a General Chair (or Co-Chair) of 38 International workshops and conferences.

A.2 Journal Paper. Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal

Table A2. JCR-Indexed Paper Information

Title	Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal
Authors	Santiago Egea Gómez, Belén Carro Martínez, Antonio Sánchez-Esguevillas and Luis Hernández-Callejo
Journal	Computer Networks (IF: 3.030)
Volume	Volume: 127, 9 November 2017, Pages 68-80
Publication Date	9 August 2017
DOI	10.1016/j.comnet.2017.07.018

Ensemble Network Traffic Classification: Algorithm Comparison and Novel Ensemble Scheme Proposal

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

ARTICLE INFO

Article history:

ABSTRACT

Network Traffic Classification (NTC) is a key piece for network monitoring, Quality-of-Service management and network security. Machine Learning algorithms have drawn the attention of many researchers during the last few years as a promising solution for network traffic classification. In Machine Learning, ensemble algorithms are classifiers formed by a set of base estimators that cooperate to build more complex models according to given training and classification strategies. Resulting models normally exhibit significant accuracy improvements compared to single estimators, but also extra time cost, which may obstruct the application of these methods to online NTC. This paper studies and compares the performance of seven popular ensemble algorithms based on Decision Trees, focusing on model accuracy, byte accuracy, and latency to determine whether ensemble learning can be properly applied to this modeling task. We show that some of the studied algorithms overcome single Decision Tree in terms of model accuracy and byte accuracy. However, the notable latency increase hinders the application of these methods in real time contexts. Additionally, we introduce a novel ensemble classifier that exploits the imbalanced populations presented in traffic networks datasets to achieve faster classifications. The experimental results show that our scheme retains the accuracy improvements of ensemble methods but with low latency punishment, enhancing the prospect of ensembles methods for online network traffic classification.

1. Introduction

In the age of the Internet, vast amounts of devices are interconnected continuously exchanging information through data networks. The exponential growth of network traffic hinders Internet Services Providers (ISPs) to manage their infrastructures efficiently and Network Traffic Classification (NTC) plays a crucial role for this task. Traffic monitoring has attracted the attention of many researchers, and Machine Learning (ML) has shown to provide successful solutions in this area [1], [2]. NTC allows network administrators to reallocate resources (e.g. underutilizing links capacity) and reconfigure network parameters (e.g. disable or enable firewall ports) to prevent Quality of Services (QoS) decays or to react to malicious behaviors [1]. As inspecting all connection flows manually is certainly unfeasible, many researchers have endeavored to develop techniques for effective NTC [2]. Network traffic classifiers aim to automatically identify traffic applications that are being used at a given instant.

NTC has to be carried out accounting for several requirements, traffic classifiers must accurately identify connection flows but, in real time conditions, there are other crucial aspects as:

- **Scalability.** Traffic classifiers will be implemented in network devices where huge amounts of packets from different users go through, so scalable classifiers are needed to manage these amounts of information [2].
- **Memory Resources.** Due to memory limitations in network nodes, classifiers must only store the most relevant information to classify applications correctly and drop variables that are not useful [2].
- **Latency.** Identification process must be as fast as possible to determine applications that correspond to each flow before it ends or an anomaly event causes QoS flaws in the network [1].
- **Privacy.** Privacy policies force network traffic classifiers not to use sensitive information obtained from users. This fact limits considerably the available information for NTC [1], [2].

Many research lines have arisen in NTC since this discipline emerged. The earliest traffic classifiers were based on the port number used by each application [1], [2]. Since port-based tools only observe port numbers used by each connection flow without any information storage, they are the simplest and fastest classifiers. However, emerging applications have no fixed ports or use different port numbers while they are running, deteriorating the accuracy of port-based classifiers. Deep Packet Inspection (DPI) tools have appeared to overcome the former limitations [2]. DPI tools inspect packet payloads in order to check byte strings for matches with prefixed patterns. DPI based approaches give accurate results, but they also have critical limitations. DPI tools need to store packet contents and inspect them, thereby their memory consumption and latency increase excessively. Additionally, databases, which contain patterns associated with each application, must be maintained and updated with zero-day applications. The maintenance of these databases is quite arduous owing to the vertiginous increase in the number of Internet protocols and applications, and encrypted traffic also complicates pattern inspections. Finally, privacy policies constrain capacity of third parties to carry out lawful deep packet inspection [1]. In this line, ML is opening the ways to develop sophisticated network traffic classifiers, which achieve an acceptable tradeoff between computation complexity and accuracy respecting users' privacy.

In ML, ensemble algorithms are complex structures formed by sets of single estimators, called base estimators, which cooperate with each other according to training and classification strategies. A large number of studies have revealed the advantages of these methods in many diverse areas and this paper aims to assess the suitability of these algorithms for NTC. Since Decision Tree algorithms are one of the most suitable learning algorithm for online NTC [1], [3], [4], this work focuses on ensemble algorithms based on Decision Trees (DTs). Despite of their high computational complexity compared to single estimators, ensemble methods may provide more accurate predictive models. As no study of clear ensemble learning for online NTC has been provided yet, seven of the most popular ensemble algorithms are compared focusing on their capabilities to be applied to this issue in this paper. We evaluate classification accuracy metrics, but also assess the computational load of each candidate. The experimental results show that ensemble algorithms exhibit higher training and classification times than a single DT, which could obstruct their implementation in real time classifiers. As possible solution, we introduce a novel ensemble scheme that consists of a sequential chain of DTs, each DT acts as connection flow filter of its successor avoiding unnecessary and repetitive classifications to decrease training and classification times.

The remainder of the article is organized as follows. Section 2 reviews relevant previous works in NTC. The methodology followed to perform our experiments and our ensemble algorithm are described in Section 3. We present and discuss the results obtained in each experiment in Section 4. Finally, the relevant conclusions of this work are presented in Section 5.

2. Related Work

The last trendy applications in NTC are based on ML algorithms. The fast-paced developments in ML have encouraged to research on these techniques in a wide number of research areas, an illustrative case is the use of clustering algorithms and Neural Networks for forecasting electricity demands [5], [6]. Although several challenging issues must be overcome yet to accomplish efficient network traffic classifiers [2], ML provides promising results for online NTC. Internet traffic identification based on ML consists of various processes, learning algorithms are trained using knowledge, which is previously acquired from captured network traces and recorded on a dataset. Dataset construction is a complicated process that dramatically affects the accuracy of traffic classifiers and their computational complexity. In online contexts, packet acquisition, training and classification times are prominent to get feasible classifiers. The main reasons why ML algorithms are excellent candidates as core of modern NTC systems are their capability of identifying network traffic respecting users´ privacy rights, their ability to handle encrypted traffic and also their capacity to be less computationally weighted than DPI tools retaining acceptable accuracies [1], [2].
Two leading learning approaches are distinguished in ML: supervised and unsupervised learning. The main difference between both approaches is that supervised learning requires a labeling process using prior knowledge about the problem in order to establish a ground truths for each connection flow; whereas training unsupervised algorithms do not need to assign application labels to each flow, and they are able to cluster classes automatically. Some researchers have adopted one of these perspectives for NTC, but hybrid techniques, known as semi-supervised approaches, have been also applied showing interesting results. This work focuses on supervised learning, namely in DTs-based algorithms.
The first relevant works in NTC [7], [8] demonstrated that application flows can be accurately identified by computing statistical attributes using few packets when connection flows start, introducing the concept of early stage classification. The authors used the first packets of TCP flows to compute instances, and they trained classifiers based on clustering methods. Although promising results were reported in terms of accuracy, they did not assess training and classification speed of their proposals. More recently, [9] has studied the efficient number of packets to perform early application identification. The authors used packet-size-based features extracted from bidirectional flows to train standard ML algorithms, including some ensemble algorithms also considered in this work (ADA Boosting and Bagging algorithm). They concluded that the optimal number of packets to correctly classify TCP flows is 5-7 and it depends on network environments. Also [4] studied how many packets could be considered to classify internet applications. They used 12 features to identify encrypted flows and compared C4.5 DT to ADA Boosting algorithm with C4.5 DT as base estimator, only one ensemble algorithm is considered in this work.
The earliest comparison among supervised learning algorithms for NTC was carried out by [3]. They compared performances between standard algorithms: Bayesian Network, C4.5 decision tree, Naïve Bayes and Naïve Bayes Trees. Furthermore, they showed that Feature Selection (FS) algorithms based on correlation measures, such as Consistency-based FS, are more suitable for NTC datasets than other approaches. Also, they found that C4.5 Decision Trees exhibited the best performances in accuracy and classification speed. [3] is one of the earliest studies that compares computational costs of ML algorithms for NTC. Later, [10] evaluates classifier performances focusing on Accuracy and Recall scores, and also on classification rate and build time (or training time). Additionally, [10] observed the influence of the composition of training data and the effect of configuring dynamic-port applications on algorithm accuracies. They trained Bayesian Networks and DT algorithms showing that sample composition of training dataset affects considerably classifier performances. Finally, they discussed the importance of labeling correctly connection flows.
Many authors have developed sophisticated ML classifiers to solve open issues in NTC. In [11], the authors combined weak learning algorithms to get more accurate predictive models, this classifier is a clear example of ensemble algorithm. Furthermore, they showed that differences among network scenarios affect classifier performances (type of applications and protocols detected, traffic distributions, link capacities and so on). Another example of ensemble algorithm is presented in [12] exploiting Sub-Space Clustering, Evidence Accumulation and Hierarchical Clustering concepts. In this work a semi-supervised approach is presented to create applications groups using clustering algorithms and assign network services labels to unknown connections in a supervised fashion. In order to create robust application groups the authors combined several clustering models using different partitions of the same dataset. A Flow-level ML classifier scheme is presented in [13], the authors designed a modular architecture for High-speed links traffic classification using m ensemble classifiers. Namely, they used OneVsRest strategy, which is also considered in this paper, but they did not assess latency of their proposal. In [14], a Robust Network Traffic classifier is presented whose main goal is to identify zero-day applications. The authors provided a parameter optimization process and compared their algorithm to Random Forest, correlation-based classification, semi-supervised clustering and Support Vector Machines, showing that the Robust Traffic Classifier overcomes other approaches. In [15], the authors proposed a self-learning classifier that starts with small number of training instances and retrains itself to improve the model performances. They implemented a decision maker to extend the number of samples in training datasets using Random Forest algorithm. Accuracy improvements were reported in each retraining iteration. Other important open topic in NTC is the effect of subflow sampling over classifier performances, which is discussed in [16] and [17]. Additionally, Naïves Bayes, Bayesian Neural Networks and Support Vector Machines algorithms are independently studied in [18], [19] and [20]. For a more general literature revision of Internet Classification area we suggest [21] and [22]. In [21] the authors review operational aspect of traffic analysis and its state-of-the-art, finally they discuss and compare relevant contributions for internet application identification, including DPI and port-based techniques; and several classification approaches are reviewed and compared using seven different network traces in [22].
Although some previous works evaluated ensemble classifiers for NTC ([4], [11], [14], [15]), none of them has compared standard ensemble methods focusing on both, accuracy and latency performances. This work tries to fill this gap by comparing ensemble schemes through several experiments assessing accuracy and latency. Additionally to ensemble algorithms comparison, a novel ensemble scheme is presented to reduce training and classification times while retaining accuracy improvements of ensemble learning respect to single DTs. For more generality, our experiments have been performed using network traffic captured in two quite different environments, three traces were captured in an ISP backbone network and the other three were captured in host computers simulating human behaviors. Below we present the methodology followed in the experiments.

3. Methodology

This section describes the methodology used in the experiments and presents the ensemble algorithms considered (Section 3.4), as well as our ensemble scheme (Section 3.5). All programs used in this paper were developed using *Python2.7*, the library *Scikit-Learn* implements the ML algorithms studied and the network traffic traces were processed using *Scapy*. *Scikit-Learn* is a well-known ML library maintained by hundreds of users and whose usage is spreading over numerous research communities. Although other tools are preferred in production due to their lesser computational complexity, this library is a suitable choice for experimental and prototyping tasks as ours.

3.1. Datasets

For our experiments we have collected six network traces captured in two quite different environments. The Internet traffic that goes through different networks differs notably between environments in the type of applications found and their distribution, it depends on the usage of network services by users and on type of entity that is serviced (enterprises, educational institutions, private houses and so on). Imbalanced label distributions in datasets significantly affect the performances of learning algorithms [23]. Thus, using several traffic traces extracted from different network environments helps to get a better understanding of the performances of traffic classifiers. Next, we introduce the network traces employed.

3.1.1. ISP traces

The ISP traces were shared by an organization that provides Internet connection to more than two millions of users across Spain. The network traffic was captured at a node in the ISP network backbone where traffic rates of 7 GB/s are supported at high load hours. Tcpdump was employed for capturing data through a port mirror for redirecting network packets and each trace lasts approximately five minutes. The processing of these traces was performed respecting privacy rights of users in a server enabled for this purpose. These traces have been captured recently, thus the presence of encrypted applications and the latest protocols is ensured. At the request of the traces providers, the name does not appear explicitly in this work due to privacy concerns.

3.1.2. HOST traces

Privacy policies obstruct the possibility of sharing network traces with the application layer from institutions or ISPs, and traces without application layers can be labeled exclusively using Port-Based tools with low trust. Due to the difficulty in getting appropriate network traffic traces, we have used three network traces manually generated in three different hosts under a controlled environment. These traces have already been used in others works to validate DPI tools [24]. The information about the network captures is shown in Table 1.

Table 1. Network Traffic Traces Information

	Start date	Duration	Datasize	# Packets	# Flows
ISP-1	17/01/2017	298 seconds	12.12 GB	8863530	231137
ISP-2	25/01/2017	259 seconds	16.96 GB	12293836	266165
ISP-3	25/01/2017	280 seconds	17.64 GB	12966391	307605
HOST-1	25/02/2013	~59 days	9438 MB	5062825	121293
HOST-2	25/02/2013	~32 days	22 GB	21000000	245627
HOST-3	25/02/2013	~65 days	7113 MB	7203000	744814

3.1.3. Attributes generation, Feature Selection and labeling process

An ad-hoc developed tool of our own was developed to extract the datasets to feed the ML algorithms from the network traces. Our software takes as input network captures stored in pcap files and the number of packets to be considered to compute the statistical attributes. Our tool is able to split initial pcap files in traces that contain packets associated with each bidirectional connection flow. Once each flow is completely stored in its corresponding trace file, they are processed to compute instances with their associated application label. The output is a dataset that contains 77 statistics regarding number of packets, packets sizes, inter-arrival packet times, TCP windows and so. The whole collection of attributes is presented at the end of this paper in Annex 1 and it includes statistics accounting for outgoing, ingoing and both directions of flows. In our experiments only the first five packets at the beginning of each flow were used to compute all statistical attributes. Because correlation-based filters have proven to be a proper FS algorithms in NTC, we have applied one of these algorithms in order to reduce the attribute space for all datasets.

For label assignment, we used a DPI tool called nDPI [25], publicly available in [26]. NDPI is able to handle encrypted traffic and is one of the most accurate open source DPI applications [27]. This tool identifies web services, as YouTube or Google, along with an extended number of protocols. However nDPI was not capable of labeling all flows in our traces and some flows were labeled as unknown. Unknown flows were depreciated in this work, since applications marked as unknown could not be determined with certainty. In other cases, some encrypted flows were identified as SSL, and port-based information was examined to distinguish between HTTPS traffic and others, as encrypted SSH connections. Both, UDP and TCP flows, were employed in our experiments. Finally, different applications and protocols were detected among the six network traces; and each application was mapped to an application group according to its protocol properties and purposes, except DNS and NTP. The application grouping was carried out according to the following protocol types: P2P includes applications as eMule, BitTorrent or eDonkey; WWW includes all HTTP and HTTPS queries to Google, Facebook, GMail and other websites; INT (INTeractive) includes protocols as SSH, Telnet, RDP and so on; Services & Control (S/C) includes network control protocols and other services as NetBios, Radius, Kerberos and so forth; Bulk includes FTP and similar protocols; Media traffic includes RTP, Skype and so on; and DB includes MsSQL, MySQL and more database applications. Other applications, as email protocols, were detected and also depreciated from this study due to their low populations.

Table 2 shows the traffic distributions found in our datasets after label assignment. Note that the network traffics are highly imbalanced according to the percentage of instances per class (%I) for the two network environments under study. In the instances of ISP-1,2 and 3, more than 70% of the samples belong to WWW traffic, and the absence of P2P, Media and Bulk traffics may be due to its restricted used at educational environments. Additionally, lower accuracy performances are expected for ISP traces owing to the point of capture. In the middle of networks, packets statistics suffer from degradation due to multipath routing, packet loss and packet duplication. Also HOST datasets are highly imbalanced, note that P2P, INT and DNS are the predominant application flows in ISP-1,2 and 3 respectively. Non-uniform sample distributions, which generally characterized NTC datasets, are exploited by the novel ensemble scheme proposed in this work to achieve faster classification and training times. Finally, note that the number of samples corresponding to Bulk and Media flows are scarce in HOST datasets, nevertheless they have an important impact on network resources due to the bytes they produce to run.

The initial datasets were subjected to a Feature Selection (FS) process. Since correlation-based FS algorithms have found effective for NTC [3], three versions of the Fast Correlation-Based Feature selection algorithm [28], [29] have been implemented for this process. As output the FS algorithms return rankings of attributes ordered according its relevance for the modelling task. These algorithms have been made publicly available in at [30]. Namely, we employed the FCBFiP version since it yielded the most accurate models in the preliminary results using the lesser number of attributes.

As final step, the datasets generated were split in training and validation subsets via a stratified technique to preserve the percentages of class populations in the training and validation phase. The 70% of the samples were used for training and the rest for validation. Next, parameters of base estimators were set using 10-fold stratified cross validation excluding the validation samples from this process, and no resampling techniques were employed avoiding to alter the class populations. This process was repeated for each iteration of the experiments.

Table 2. Datasets Information. %I denotes the percentage of instances in the dataset and %B the percentage of Bytes in the network capture

	P2P		WWW		DNS		INT		S/C		Bulk		Media		NTP		DB	
	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B
ISP-1	0.17	<0.01	80.46	99.60	16.28	0.08	1.99	0.10	0.73	0.24	-	-	-	-	0.37	<0.01	-	-
ISP-2	-	-	75.30	99.60	21.50	0.10	2.52	0.12	0.35	0.18	-	-	-	-	0.34	<0.01	-	-
ISP-3	-	-	71.70	99.60	24.98	0.11	2.66	0.12	0.37	0.22	-	-	-	-	0.29	<0.01	-	-
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	-	-	-	-
HOST-3	2.01	38.10	8.31	39.06	79.81	4.05	4.94	2.58	-	-	0.68	6.01	0.42	9.38	3.73	0.79	0.10	0.03

3.2. Evaluation Environment

All experiments were performed in a workstation with 12GB of memory RAM and CPU AMD A10 6800K (4.1Ghz). Although the CPU has four cores and *Scikit-Learn* allows to train models in parallel, we used only one for our experiments in order to isolate each experiment in a unique processing core. As decision trees are sensitive to random initializations at the beginning of their learning phase, they suffer from variance

when they are trained. To diminish the effects of models variance, the experiments were repeated ten times and the mean of resulted metrics is reported in Results section.

3.3. Performance Metrics & Statistical Validation

We have studied several metrics to compare the proposed algorithms and applied a statistical validation procedure over the results as we describe below. Model performances were assessed isolating completely the validation datasets from the training processes.

3.3.1. Overall Accuracy

Overall Accuracy (OA) is the percentage of samples labeled correctly. In this way, OA is defined as

$$OA = \frac{\sum TP_i}{\#Samples} \quad (1)$$

Where TP_i denotes True Positives associated with the class i and #Samples denotes the number of samples contained in the datasets.

3.3.2. Class Accuracies

Because OA is the percentage of samples correctly labeled and network traffic is highly imbalanced, great precisions over high populated traffic will hide errors on application flows with low populations in the datasets (Table 2). Therefore we included the individual accuracy for each class. Thus, we define the Accuracy for a given class i as

$$A_i = \frac{TP_i}{\#Samples\ of\ class\ i} \quad (2)$$

Where $\#Samples\ of\ class\ i$ is the number of samples associated with the class i.

3.3.3. Byte Accuracy

OA and Class Accuracies alone could be insufficient to assess model performances, it is interesting to study how many bytes have been accurately labeled to appreciate more clearly algorithm reliabilities. Thus, as it is an insightful metric in NTC, we have included the Byte Accuracy metric (BA) in our comparison, BA is defined as

$$BA = \frac{Bytes\ labeled\ correctly}{Total\ bytes\ captured} \quad (3)$$

3.3.4. Number of Features used in the models

As scalability and latency are important properties for online NTC classifiers, we have included in our results the number of statistical attributes used by each algorithm. Furthermore, we provide a model complexity evaluation to determine if ensemble algorithms are able to equal or overcome a single estimator performances using more reduced subsets, and thus, bringing computational benefits in the attribute computing phase.

3.3.5. Training and Classification times

Finally, we measured Training and Classification times to quantify computational punishment of using different ensemble schemes. Although Classification time is more prominent in online NTC, novel classifiers include retraining phases, therefore, Training times are also relevant in our comparison.

3.3.6. Statistical Validation: Friedman´s Test

Since the average of measurements obtained from experiments using different datasets might sometimes be insufficient to validate general observations, we conducted a statistical validation process to make our results more rigorous [31], [32]. After measuring the previous properties for each algorithm over the six datasets studied, we have applied a well-known statistical method to compare multiple algorithms, the Friedman´s test. The Friedman´s test is a non-parametric statistical method for detecting differences amongst more than two related experiments. This procedure ranks the compared algorithms according to their results obtained for each dataset. The best scored algorithm is assigned the value 1 and the worst scored gets the value k, being k the number of compared algorithms. The Friedman´s test is computed by equation (4), where R_j is the average score for the algorithm j over the N datasets, with $N = 6$ for this case.

$$\chi^2{}_F = \frac{12N}{k*(k+1)}\left[\sum_j R_j^2 - 0.25k*(k+1)^2\right] \quad (4)$$

Once $\chi^2{}_F$ is computed, the p-value is obtained from a chi-squared random distribution with $k-1$ degrees of freedom. We have set the significance threshold at $\alpha = 0.05$. Thereby if the p-value is lesser than α, the null hypothesis, which states that statistical difference amongst candidates does not exist, is rejected.

In addition to the Friedman´s test, we have applied a post-hoc correction method called the Holm´s procedure [32]. This method uses the adjusted p-values (APVs) to compare the performance of a control algorithm with respect to the rest, normally the control algorithm is the best scored in the Friedman´s ranking. The algorithms are sorted according to their average scores R_j, and the associated APVs are computed as $APV_j = \alpha/(k-p)$ where p is the position of the algorithm j in the ordered ranking. The value z_j is computed for each algorithm using equation (5), where R_i is the average score of the control algorithm in the Friedman´s test. The value z_j follows a normal distribution and its associated probability p_j can be obtained evaluating $Z(z_j)$. If $p_j < APV_j$, we conclude that a significant difference exists among the algorithm j and the control algorithm.

$$z_j = \left(R_i - R_j\right)/\sqrt{\frac{k(k+1)}{6n}} \quad (5)$$

This statistical validation procedure is similar to the methods applied in [9]. For more information about these methods read [31] and [32].

3.4. Ensemble Classifiers

Ensemble classifiers are learning algorithms composed by multiple base estimators along with training and classification strategies to make final decisions [33]–[39]**.** Since DTs yield satisfactory results in NTC ([1], [3], [4]), we have selected the CART DT algorithm, provided by the *Scikit-learn* library, as base estimator for the ensemble structures. Ensemble algorithms can be distinguished according to the training and classification strategies they employ. *Scikit-learn* library contains a wide number of popular ensemble algorithms, some of these algorithms have been considered for the experiments presented in this paper. Below, we briefly describe the ensemble algorithms selected.

- **OneVsRest.** One classifier is built per class to distinguish one class from the rest [38], thereby one dataset is generated for each class to train each base estimator. Finally, unknown samples are classified according to the estimate of the posterior probability for each class: given an unknown sample, the class whose posterior probability is maximum is assigned to that sample.

- **OneVsOne.** One base estimator is trained to distinguish between two different classes excluding the rest from the training. Therefore, $n(n-1)/2$ datasets and classifiers are built for n classes. The final label is assigned by majority voting: the most voted class amongst all classifiers is the class associated with the unknown sample [38].
- **Error-Correcting Output-code (OutputCode).** One binary code is associated with each class and one classifier is trained in parallel per each bit. In classification, a new instance generates a code that is projected onto the binary space, and the closest label to the projected point is assigned to the unknown sample [36]. The code size is a design parameter that determines the number of classifiers in the model, 12 base estimators were used in this work.
- **Adaboost classifier (ADA).** This algorithm is composed by a set of weak estimators that are trained sequentially and a set of weights associated with each class [33], [34]. In each training iteration, misclassified classes are awarded by increasing their associated weights. In contrast, classes with less error rate are punished decreasing their weights. Adaboost implements training and reweighting phases in its training process that speed it down considerably. Finally, Label assignment is performed via weighted majority voting. The number of estimators were set to 20 for Adaboost in the experiments to reach a right tradeoff between latency and accuracy.
- **Bagging algorithm.** In this instance a large number of base estimators are trained in parallel using different datasets [33], [40]. Each dataset is generated applying bootstrap resampling and is used to train only one classifier. Majority voting strategy is used for label assignment. We set the number of base estimators to 20, since including many estimators leads to low classification and training speeds, whereas a low number of base estimators could lead to poor accuracy.
- **Random Forest (RF).** RF is a combination of several DTs, whose training process is based on the generation of random subsets from the original dataset to feed each DT [34], [37]. Unlike Bagging, each subset is built by random selection of samples and attributes. The final label assignment is based on majority voting. Such as ADA and Bagging, the number of trees in the forest were set to 20.
- **Extremely Randomized Trees (ExtraTrees).** This algorithm is very similar to Random Forest but with two differences: ExtraTrees does not generate new training datasets but instead it uses the initial one; and also it does not choose the best splits, but chooses the split randomly [39]. Such as the former algorithms, the number of trees were set to 20.

Finally, our proposed ensemble algorithm was considered for the comparison. We describe this novel proposal, called Tailored Decision Tree Chain (T-DTC), in the following section.

3.5. Our proposal: Tailored Decision Tree Chain

As it is discussed in Section 3.1, network traffic is highly imbalanced (Table 2), e.g. DNS traffic is quite more populated than Bulk, S/C and others in our datasets. This fact is not exclusive of the datasets used in this study, network traffic has been studied by several researchers showing that traffic distributions are highly imbalanced in many environments [41], [42]. Furthermore, some type of flows are easier to identify than others as our experimental results show. These facts could be exploited to reach more efficient ensemble classifiers for online NTC. Next, a novel ensemble algorithm, called Tailored Decision Tree Chain (T-DTC), is introduced.

3.5.1. Ensemble Scheme & Classification process

Our proposal is based on the use of the fastest and most accurate DTs to classify and filter out samples avoiding repetitive classification of instances that are easily identified. In our scheme, a set of classifiers are sequentially ordered as a chain and trained to distinguish one traffic application from the rest, so that when T-DTC assigns an application label to an unknown sample, the connection flow is filtered out from the classification process and it is not classified in later stages. On the contrary, when T-DTC assigns the label "other" to an unknown sample, the instance passes to the ensuing classification stage to check if the sample corresponds to other application flow in the chain. This process is redundant until an application is assigned to the unknown sample and, immediately after the flow is identified, it is output from the classification process. Figure 1 depicts this idea for the classes contained in ISP-1 once an appropriate order of classifiers was determined by the procedure described in next section**.** Note that more than 80% of the network traffic is classified by the first stage requiring being identified only by one DT; above 95% of flows are identified in the following two stages and roughly the 2% of the instances reach the two last classifiers passing through all classification stages. Note also that this scheme requires only $n-1$ classifiers (where n is the number of application flows to identify), since the last two classes share the same DT. This approach requires less classifiers than other strategies that are included in this paper (e.g. OneVsRest is composed by n classifiers, and OneVsOne uses $((n-1)*n/2)$) consuming less memory resources than other ensemble schemes. Next, we present the procedure followed to determine the proper order of DTs into T-DTC.

3.5.2. Ordering the classifiers

The order of the classifiers in the chain is crucial, since if inaccurate classifiers are put at the beginning of the chain, misclassified samples will not reach their corresponding DT and accuracy performances will diminish drastically. Thus, the fastest and most accurate DT must be put in the first classification stages. As the number of combinations grows exponentially as more type of applications to identify and testing all combinations is computationally weighted, we have studied the error metrics amongst classes to correctly order the classifiers in our structure. For that purpose, we trained a single DT and inspected the confusion matrix using only the training dataset for each network capture. This process considerably reduces the number of combinations considered as proper orders resulting in a bound set of choices. Finally, the order of the classifiers was chosen by assessing OA among the possibilities via cross validation.

The best order for the six datasets using this procedure were: WWW-DNS-NTP-INT-S/C-P2P for ISP-1 (as Fig 1 shows); WWW-DNS-NTP-INT-S/C for ISP-2; WWW-DNS-INT-S/C-NTP for ISP-3; P2P-S/C-WWW-DNS-INT-Bulk-Media for HOST-1; INT-P2P-DNS-WWW-Bulk-Media-S/C for HOST-2; and NTP-DNS-INT-P2P-WWW-DB-Bulk-Media for HOST-3.

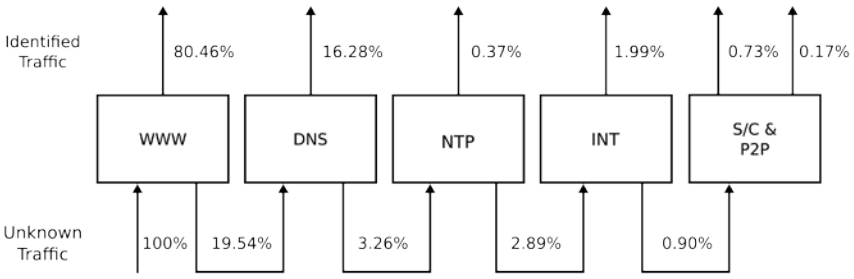

Figure 1. Tailored Decission Tree Chain Structure for ISP-1 once the classifiers were ordered

3.5.3. Training process

Considering n traffic applications, our algorithm generates $n-1$ datasets from the initial one to train each DT. For example, according to Fig 1 T-DTC needs to input the whole training dataset to the first classifier, but reassigning the labels different from WWW to "other". In the instance of the second classifier in the chain, as WWW traffic has been identified in the previous stage, the samples belonging to this traffic application have to be removed from the dataset; and samples that do not belong to DNS traffic are labeled as "other". This process is repeated until reaching the last classifier, which do not need label reassignments. Once all datasets are generated, classifiers in chain were trained in similar way to other schemes, as OneVsOne, OneVsRest or OutputCode.

4. Results

In this section, we discuss the experimental results obtained from comparing the ensemble algorithms, including our proposal, to a single DT. In order to show a clearer comparison of the ensemble algorithms, we have remarked the model that provides the best OA score for each algorithm varying the size of the subsets according to the attribute ranking provided by the FS algorithm. We present and discuss accuracy metrics in Section 4.1, and computational time during training and classification phases in Section 4.2**;** and later, the results presented are undergone to a statistical validation procedure in Section 4.3. Below, we discuss model complexity in terms of number of statistical attributes that each classifier has to compute to accurately classify Internet traffic. In Section 4.4, we evaluate how many attributes at least each ensemble algorithm need including in its training phase to outperform or equal the best DT models. Through this experiment, we find out the models that provide better performances than DT using the less number of statistical attributes. Finally, we provide a summary of our results in Section 4.5.

4.1. Overall Accuracy and Byte Accuracy Evaluation

Table 3 contains the results obtained using the three datasets captured in the ISP backbone and Figure 2 depicts graphically the class accuracies obtained for all network traces. Figure 2 is a colormap that represents the accuracies obtained for each traffic class detected in the six network traces. The horizontal axis contains the applications found in each network traces, meanwhile each row of the vertical axis corresponds to each algorithm. In the case of ISP-1, we observe that the best results in terms of Overall Accuracy (OA) were provided by T-DTC, followed by OutputCode and OneVsRest. These three algorithms improve the accuracy for high populated traffic, WWW and DNS, resulting in higher OA scores. Ensemble algorithms generally overcome a single DT with the exception of OneVsOne and ExtraTrees. Although ADA yields high accuracy scores for WWW and DNS traffics, its performances over the rest of traffic applications are very poor affecting negatively the OA score. Examining the Byte Accuracy scores (BA), we find that the highest performances are provided by T-DTC, Bagging and OutputCode. In general, the learning algorithms yield similar results for the three ISP datasets, the three highest OA and BA for ISP-2 were provided by T-DTC, OutputCode and OneVsRest. Finally, the most accurate models in terms of OA for ISP-3 were T-DTC, Outputcode and OneVsRest; and, in terms of BA, OneVsRest outperforms OutputCode and T-DTC remains as the best score.

Table 4 shows the accuracy scores obtained for the network traffic captured simulating host activity artificially. The best algorithm for HOST-1 in terms of OA was Random Forest, followed by Bagging and Extremely Randomized Trees. Observing Table 4 and Figure 2 we find that the greater accuracies identifying the high populated classes (P2P, WWW and INT) compared to a single DT result in OA improvements for these three algorithms. In general, the OA using a single DT is improved by most of ensemble algorithms, only ADA Boosting and OneVsOne got worse OA for HOST-1. If we focus on BA the observations change, the winner algorithm is OneVsOne nearly followed by ExtraTrees and RF. Finally, ADA and T-DTC yielded poor byte accuracies due to accuracy diminishing for Bulk and Media flows. In the instance of T-DTC, the accuracy loss was caused by error propagation between classifiers in the chain, when an application flow is misclassified in the first stages, T-DTC will yield low accuracy for that; error propagation is discussed below in this section. For HOST-2, the three most accurate models in terms of OA were provided by ExtraTrees, RF and OutputCode, respectively; while ADA yielded very poor results as in the former network trace. The accuracy improvements for the predominant application flows, especially WWW, result in the most of ensemble algorithms overcoming single estimator OA. Observing Byte Accuracy, none of the ensemble methods overcomes a single DT, the only ensemble algorithm that provides Byte Accuracy near to DT´s are T-DTC and OneVsOne for HOST-2. This BA reduction is owing to the fact that most of the ensemble methods yield worse results than a DT for Bulk and Media traffic, whose impact over the byte distribution is decisive (see Table 2). In the case of T-DTC, the accuracy loss over Bulk and Media is offset by a better interactive (INT) traffic identification. In the instance of HOST-3, we can observe that the greatest OAs were provided by RF, OutputCode and T-DTC. Whereas the best BAs were resulted from training T-DTC, ExtraTrees and OutputCode, respectively. Although in terms of OA the ensemble models do not seem to provide significant improvements for HOST-3, the BA for these algorithms is notably better than DT´s.

Comparing the results obtained in the two network environments, we can observe that ensemble learning generally provides better results in terms of Overall Accuracy and Byte Accuracy than a single DT for ISP and HOST traces. Only ADA algorithm exhibited worse performances than a single DT for most of cases studied making its application to NTC not recommended. Also, we can observe that perfect ensemble algorithm that performs clearly better than other candidates for all traces is not found. However, T-DTC performances are the best or very close to the best for all traces except HOST-1. The excellent performances achieved in most cases is due to when the OA and Class Accuracies are higher for a given dataset, less errors propagate from the first classification stages of T-DTC to following stages, meaning that consistency of labels contained in datasets is a determinant fact for T-DTC. Unlike HOST-2,3 and ISP-1,2,3, T-DTC suffers from deterioration of BA when it is trained with the dataset HOST-1. This performance decay is due to the poor Bulk and Media accuracies for this dataset, note that Bulk and Media traffic populate an important percentage of Bytes in this network capture (see Table 2), and thus the errors committed over these traffic flows have a major influence on the general performances of classifiers. Other remarkable observation is that when an application flow is more populated in a NTC dataset, classifiers normally exhibit better performances on it than other traffics, being class distributions an important fact for NTC. Finally, a substantial difference is found between the datasets obtained from a host (HOST-1,2,3) and datasets obtained from ISP (ISP-1,2,3). Due to the fact that ISP-1,2,3 were captured at a point placed in the middle of the backbone, the statistical attributes have higher variance hindering the general performances in terms of accuracy metrics, especially for P2P, DNS, S/C and INT flows. Note that high populated classes in HOST and ISP traces, as WWW and DNS, are more resilient to high variance of the statistical attributes. Furthermore, ISP traces are more cutting edge, and consequently, the presence of encrypted connections is higher than in HOST traces complicating the identification of application flows, as WWW or INT, which permit the use of encryption protocols.

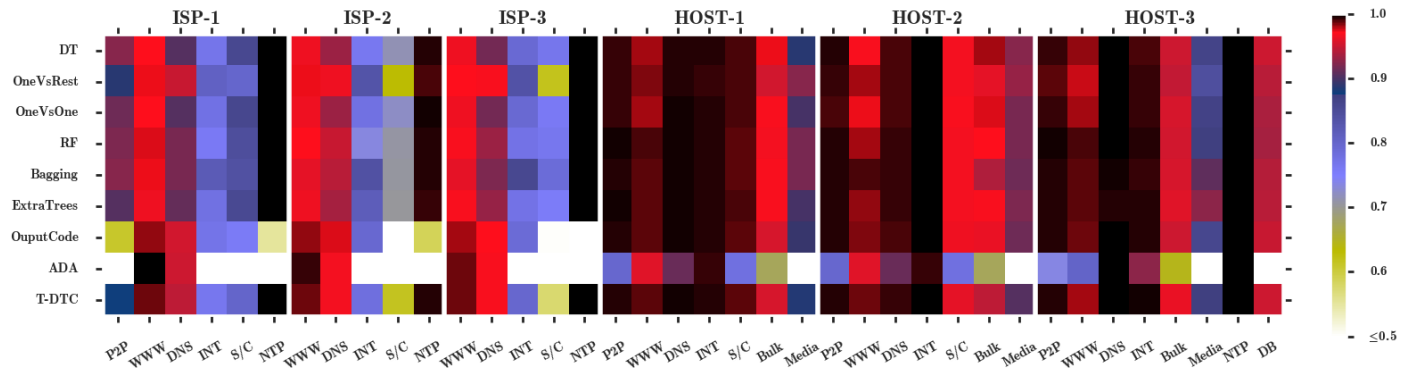

Figure 2. Class Accuracies for the algorithms and network traces studied

Table 3. General performances for ISP traces after applying Feature Selection

	OA	BA	# Features
ISP-1			
DT	0,95685	0,99417	9
OneVsRest	*0,96502*	0,99850	11
OneVsOne	0,95680	0,99445	8
RF	0,96081	0,99586	8
Bagging	0,96122	0,99897	10
ExtraTrees	0,95249	0,99336	27
OuputCode	0,97011	*0,99873*	19
ADA	0,95915	0,99698	6
T-DTC	**0,97400**	**0,99903**	14
ISP-2			
DT	0,95310	0,93077	32
OneVsRest	*0,96681*	*0,95995*	22
OneVsOne	0,95384	0,95169	30
RF	0,95962	0,95390	20
Bagging	0,95467	0,95002	12
ExtraTrees	0,95492	0,94860	25
OuputCode	0,97396	0,96047	28
ADA	0,95587	0,94488	13
T-DTC	**0,97604**	**0,97317**	22
ISP-3			
DT	0,94817	0,95078	22
OneVsRest	*0,96697*	0,97340	22
OneVsOne	0,94845	0,95943	21
RF	0,95434	0,96342	18
Bagging	0,94889	0,95867	13
ExtraTrees	0,95476	0,92762	14
OuputCode	0,97003	*0,97126*	23
ADA	0,95055	0,96480	13
T-DTC	**0,97603**	**0,97936**	18

Table 4. General performances for HOST traces after applying Feature Selection

	OA	BA	# Features
HOST-1			
DT	0,98652	0,94162	30
OneVsRest	0,98718	0,94407	22
OneVsOne	0,98646	**0,95824**	30
RF	**0,99041**	*0,95611*	30
Bagging	0,99005	0,95502	29
ExtraTrees	*0,98986*	0,95709	30
OuputCode	0,98840	0,94910	30
ADA	0,92842	0,59624	8
T-DTC	0,98790	0,91688	43
HOST-2			
DT	0,99192	**0,99452**	32
OneVsRest	0,99258	0,97969	20
OneVsOne	0,99194	*0,99357*	16
RF	0,99346	0,98962	12
Bagging	0,99330	0,98419	12
ExtraTrees	**0,99348**	0,98556	14
OuputCode	*0,99336*	0,98118	22
ADA	0,92842	0,89946	9
T-DTC	0,99319	0,99447	38
HOST-3			
DT	0,99742	0,95806	38
OneVsRest	0,99702	0,95518	42
OneVsOne	0,99750	0,95263	42
RF	**0,99813**	0,96934	42
Bagging	0,99590	0,96492	43
ExtraTrees	0,99363	0,97793	42
OuputCode	0,99795	*0,97057*	30
ADA	0,97526	0,87591	35
T-DTC	*0,99792*	**0,97832**	21

4.2. Time Performance Comparison

Table 5 contains the results obtained from measuring the computational times in both, Training and Classification phases, for the models included in Table 3 and 4. Since ADA yielded poor performances for OA and BA, it has been depreciated from this discussion. As expected, the fastest algorithm is a single DT compared to ensemble classifiers for all traces due to its lesser complexity.

Table 5. Training and Classification time for the traces studied (in seconds)

	ISP-1		ISP-2		ISP-3		HOST-1		HOST-2		HOST-3	
	Training Time	**Classification time**	**Training Time**	**Classification time**	**Training Time**	**Classification time**	**Training Time**	**Classification time**	**Training Time**	**Classification time**	**Training Time**	**Classification time**
DT	**0.94330**	**0.02095**	4.33370	**0.03834**	**2.85369**	**0.03146**	**1.38806**	**0.01901**	**1.64008**	**0.03180**	**4.31513**	**0.08269**
OneVsRest	5.55971	0.08686	10.55774	0.11443	10.99850	0.11432	4.65841	0.08323	5.36138	0.13081	37.53189	0.54066
OneVsOne	2.54106	0.37229	12.78923	0.52626	7.44010	0.36590	6.00742	0.52563	3.03271	0.74541	23.16181	3.45232
RF	2.51751	0.29148	7.81902	0.49980	5.71650	0.38325	3.47846	0.30827	2.63806	0.40228	13.88464	1.17011
Bagging	11.04407	0.35490	14.49463	0.42822	15.90904	0.39981	15.01821	0.37088	6.53809	0.40831	54.44654	2.02192
ExtraTrees	3.14286	0.52690	4.26417	0.61417	3.20274	0.64590	2.24429	0.53985	1.74953	0.51918	9.03618	1.76928
OuputCode	17.09089	0.18528	23.63508	0.20806	20.17624	0.19767	16.95315	0.18262	10.46566	0.23858	35.89734	0.82363
T-DTC	1.52025	0.04099	**3.87237**	0.06602	3.21352	0.05421	1.95967	0.04057	1.76991	0.06958	4.38929	0.48791

In the case of ISP datasets, T-DTC provided the fastest ensemble model in classification and training, even T-DTC exhibited lesser training time than a DT for ISP-2. OneVsRest and OutputCode are the second and third fastest ensemble algorithms in classification, although their

training phases are much longer than T-DTC´s. OneVsOne, RF, Bagging and ExtraTrees exhibited quite long classification times being more than ten times slower than a single DT.

Focusing on HOST traces, the fastest ensemble algorithm is anew T-DTC for Training and Classification phases for the datasets HOST-1 and HOST-3. In the instance of HOST-2, T-DTC is overcome by ExtraTrees in Training, although T-TDC remains providing the best Classification Time. Although ExtraTrees retains a reasonable Training Time for HOST traces, its classification phase is quite more complex resulting in long Classification Times. OneVsRest yielded the second fastest Classification Time for all HOST traces, however its training phase is longer than other classifiers that provide more accurate models (see Table 4), as Random Forest, ExtraTrees and T-DTC. The slowest algorithms in Training are Bagging and OutputCode, but in Classification they spend less time than OneVsOne and ExtraTrees.

The cost of employing ensemble algorithms in NTC is clear as it is shown in Table 5, all ensemble algorithms suffer a latency increase in Training and Classification. Although the time punishment is higher if the network capture contains more connection flows, it is very different among ensemble algorithms. Algorithms formed by a huge number of classifiers exhibited a significant increase in their times, this is the case of OneVsOne, RF, Bagging and ExtraTrees; unlike them, OneVsRest and T-DTC do not suffer from huge time increases, since they are formed by a fewer number of classifiers. Because they are composed by a similar number of base estimators (for n classes OneVsRest trains n estimators and T-DTC trains $n-1$), they spend similar times in the classification task, being T-DTC always faster due to its classification strategy and more accurate for the majority of the analyzed traces.

4.3. Statistical Validation

In this Section we present the results obtained from assessing the statistical significance of the performances presented in Table 4 and 5. Table 6 shows the Friedman´s test scores along with the p-values and APVs obtained by applying the Holm´s procedure. Note that statistical differences exist between algorithms for almost all performances with less than 0.05 level of significance. Only Byte Accuracy obtained a p-value greater than α. Although BA p-value is greater than 0.05, it is lesser than 0.1 thus retaining high relevant differences among algorithms.

For the Overall Accuracy, the three best performances are obtained by OutputCode, T-DTC and RF respectively. Setting OutputCode as control algorithm for the Holm´s procedure, we find that no relevant statistical differences exist for OneVsRest, RF, Bagging, ExtraTrees and T-DTC; meanwhile OutputCode OA differs considerably from DT, OnevsOne and ADA. Focusing on Byte Accuracy we observe that T-DTC is clearly the best algorithm followed by OutputCode and RF, and that there are not big statistical differences between ensemble methods with the exception of ADA algorithm. For the instance of Training Time, we note that DT is the fastest algorithm, as expected, and T-DTC ties with ADA and ExtraTrees. Being DT the control algorithm, we can say that there are no differences between DT and RF, ExtraTrees, ADA or T-DTC. Finally, DT is the fastest in classification, and T-DTC and OnevsRest are the second and third fastest algorithms. According to the Holm´s procedure there are not relevant differences between the previous three algorithms and DT in classification.

Table 6. Friedman´s Test and Holm´s procedure results

	Overall Accuracy (OA)			Byte Accuracy (BA)			Training Time			Classification Time		
	Ranking	p-values	APVs	Ranking	p-values	APVs	Ranking	p-values	APVs	Ranking	p-values	APVs
DT	7.50	0.001	0.006	6.50	0.008	0.007	**1.66**	-	-	**1.00**	-	-
OneVsRest	4.50	**0.205**	0.016	5.00	**0.091**	0.009	6.83	0.001	0.008	3.00	**0.205**	0.025
OneVsOne	7.17	0.003	0.008	5.00	**0.091**	0.009	6.00	0.006	0.010	7.83	< 0.001	0.007
RF	3.00	**0.752**	0.025	4.33	**0.206**	0.025	4.50	**0.073**	0.0125	5.83	0.002	0.012
Bagging	5.17	**0.091**	0.011	5.00	**0.091**	0.009	8.17	< 0.001	0.007	6.67	< 0.001	0.008
ExtraTrees	5.17	**0.091**	0.011	5.67	**0.035**	0.008	3.33	**0.292**	0.020	8.17	< 0.001	0.006
OuputCode	**2.50**	-	-	3.83	**0.348**	0.050	8.67	< 0.001	0.006	4.00	**0.058**	0.017
ADA	7.33	0.002	0.007	7.33	0.002	0.006	3.33	**0.291**	0.02	6.50	< 0.001	0.010
T-DTC	2.67	**0.916**	0.050	**2.33**	-	-	3.33	**0.598**	0.050	2.00	**0.527**	0.050
Friedman´s	25.91	0.001	-	13.64	0.091	-	40.80	< 0.001	-	43.02	< 0.001	-

4.4. Model Complexity Evaluation

We have already assessed the best models built by each ensemble algorithm. Before a connection flow is classified, the classifier has to compute the statistics associated with it, and also it may be interesting to assess how many attributes we can drop to equal or overcome the best DT performances for each ensemble algorithm. Although ensemble algorithms yield longer training and classification times, they could offer other advantages by reducing the number of statistical attributes used for training predictive models. Table 7 contains the results obtained using the best DT OA as baseline for each dataset. From Table 7, we can observe that RF, OutputCode and T-DTC provided models that overcome the baseline OA using a reduced number of attributes for all dataset. RF got the best OA for HOST-1,2 in spite of using quite reduced subsets. Although T-DTC yielded worse BA than the baseline for ISP-1 and HOST-2, its accuracy metrics were the best or almost the best using a lesser number of attributes for ISP-2-3 and HOST-1-3. Bagging also exhibited good scores with a high subset reduction, and DT only got higher BA than its competitors for ISP-2 using many more features. These experimental results show that ensemble learning algorithms are able to get similar performances to DT´s using less statistical attributes. The most of ensemble algorithms equaled or overcame single DT performances for almost all traces, only, ADA and OneVsOne achieved worse results than DT´s in four or more of the network traces. The subset reduction could offset the training and classification time punishment discussed in Section 4.2.

Finally, we provide a rank of the most relevant features resulting from applying feature selection to our datasets. To appreciate the features that more contribute to the predictive models, we have counted the number of times the features appeared in the models with less number of attributes shown in Table 7; and Table 8 ranks the statistical attributes that are included in the different models at least two times. As Table 8 shows, none of the attributes in the datasets is used for the six analyzed datasets. The most employed attributes are maxTCPWin0 and NPKT_128. Although three of the most relevant attributes depend on TCP windows, packet-size based attributes have a remarkable presence as informative features. In Internet networks, Packet-size based features are more resilient to the operational status of networks (e.g. if the network is congested) than TCP-windows based features, therefore packet-size based attributes are more interesting for NTC. More research must be conducted to determine the optimal set of statistical attributes independently to network environments and operational status of networks.

Table 7. First models in accomplishing the same Overall Accuracy (OA) as the best DT model. (BA = Byte Accuracy and # is the number of features used)

	ISP-1			ISP-2			ISP-3			HOST-1			HOST-2			HOST-3		
	OA	**BA**	**#**	**OA**	**BA**	**#**	**OA**	**BA**	**#**	**OA**	**BA**	**#**	**OA**	**BA**	**#**	**OA**	**BA**	**#**
DT	0.95685	0.99417	9	0.95310	0.93077	32	0.94817	0.95078	22	0.98652	0.94162	30	0.99192	**0.99452**	32	0.99742	0.95806	38
OneVsRest	**0.96339**	0.99456	5	0.96284	0.95744	6	0.95651	0.96294	**4**	0.98682	0.94251	19	0.99207	0.97957	19	-	-	-
OneVsOne	-	-	-	0.95370	0.93042	28	-	-	-	-	-	-	-	-	-	0.99750	0.96185	42
RF	0.95859	0.99467	5	0.95632	0.93149	6	0.95204	0.96229	7	0.98656	**0.95407**	16	**0.99260**	0.97829	**8**	**0.99799**	0.97158	38
Bagging	0.95779	**0.99502**	5	0.95467	0.94032	12	0.94888	0.95901	12	0.98667	0.95639	20	0.99234	0.98068	9	-	-	-
ExtraTrees	-	-	-	0.95335	0.90746	17	0.94883	0.89486	10	0.98696	0.95333	16	0.99286	0.97556	9	-	-	-
OuputCode	0.96162	0.99487	5	0.96549	**0.96035**	6	0.95542	0.96053	5	**0.98745**	0.94172	17	0.99243	0.97917	10	0.99753	0.96852	13
ADA	-	-	-	-	-	-	0.94982	0.96480	12	-	-	-	-	-	-	-	-	-
T-DTC	0.95808	0.95615	**3**	**0.97302**	0.93900	**6**	**0.96712**	**0.97109**	**4**	0.98662	0.94648	**11**	0.99240	0.98093	12	0.99750	**0.99205**	**12**

Table 8. Ranking of the most relevant features used for all the datasets included in the experiments

Feature Name	Description	# of times used
maxTCPWin0	Maximum TCP Windows used in flow packets considering outgoing direction.	4
NPKT_128	Number of packets whose applications bytes is higher than 64 and lesser than 128 bytes considering both directions.	4
maxBytes0	Maximum number of bytes transferred in flow packets considering outgoing direction.	3
%Bytes0	Percentage of bytes transferred in flow packets considering outgoing direction over the total number of bytes in whole flows.	3
minBytes0	Minimum number of bytes transferred in flow packets considering outgoing direction.	3
NPKT_64	Number of packets whose application bytes are higher than 32 and lesser than 64 bytes.	3
minTCPWin0	Minimum TCP Windows used in flow packets considering outgoing direction.	3
varBytes1	Variance of the bytes transferred in flow packets considering ingoing direction.	3
%Packets0	Percentage of the number packets transferred considering outgoing direction over the total number of packets in whole flows.	2
varTCPWinT	Variance of TCP Windows used in flow packets considering both directions.	2
maxBytesT	Maximum number of bytes transferred in flow packets considering both directions.	2

4.5. Summary

Finally, Table 9 contains a summary of the results discussed in previous sections. Since we intend to compare only ensemble algorithm, we exclude DT from this summary; and also ADA algorithm was excluded due to its poor performances.

Table 9. Summary of results

	ISP-1	ISP-2	ISP-3	HOST-1	HOST-2	HOST-3
Overall Accuracy (OA)						
Highest	T-DTC	T-DTC	T-DTC	RF	ExtraTrees	RF
Lowest	ExtraTrees	OneVsOne	OneVsOne	OneVsOne	OneVsOne	ExtraTrees
Byte Accuracy (BA)						
Highest	T-DTC	T-DTC	T-DTC	OneVsOne	T-DTC	T-DTC
Lowest	ExtraTrees	ExtraTrees	ExtraTrees	T-DTC	OneVsRest	OneVsOne
Training Time						
Fastest	ExtraTrees	T-DTC	T-DTC	T-DTC	ExtraTrees	T-DTC
Slowest	OuputCode	OutputCode	OutputCode	OuputCode	OuputCode	Bagging
Classification Time						
Fastest	T-DTC	T-DTC	T-DTC	T-DTC	T-DTC	T-DTC
Slowest	ExtraTrees	ExtraTrees	ExtraTrees	ExtraTrees	OneVsOne	OneVsOne

As Table 9 reveals, T-DTC is the fastest ensemble scheme in classification for all datasets studied. Also T-DTC speeds up notably the training phase getting the fastest times for four datasets and the second fastest for the others. Finally, T-DTC exhibits the highest BA for all ISP traces and two of the three HOST traces; and, in terms of OA, T-DTC yielded the best or very close to the best scores. The experimental results obtained validate our proposal for our datasets, improving the prospects of ensemble learning in real time NTC.

5. Conclusions

This work compares the performance of seven popular DT-based ensemble algorithms along with a single base estimator (DT) and a novel proposed ensemble scheme in order to assess the suitability of ensemble learning in real time NTC. Although some ML algorithms have

previously been studied in related works for this problem, they lack of a clear and detailed comparison among ensemble algorithms assessing both accuracy metrics and computational costs. Our experimental results show that most of the ensemble algorithms improve the performance metrics of a single estimator for our datasets, but, as expected, extra time costs are found in classification and training phases. All ensemble algorithms analyzed in this paper exhibit a notable classification-time increase that hinders the use of these techniques in real time contexts. Also we have shown that some ensemble schemes are able to equal or overcome a single DT using a reduced subset of attributes, leading to computational savings, which could offset classification and training time punishments. Therefore, the performance improvements and attribute reduction of some ensemble algorithms could justify their implementation in some contexts, such as small networks or environments where the computational capacity of network devices is not a crucial limitation.

With the intention of boosting ensemble learning in online NTC, a novel ensemble method, called T-DTC, is proposed. T-DTC exploits imbalanced traffic distributions in NTC datasets to significantly speed up Training and Classification phases. T-DTC achieves time savings by ordering CART Decision Trees in a sequential chain in which each base estimator is trained to distinguish only one traffic application from the rest. Thereby, each classifier filters out samples that are assigned to an application group and feeds following classifiers with samples whose application is not detected, avoiding repetitive classifications of classes that are easily and accurately identified.

In this paper, we show that our ensemble scheme clearly outperforms other ensemble algorithms in terms of latency, but also retaining the essential performance metrics (such as Overall Accuracy and Byte Accuracy) improvements of ensemble learning respect to a single Decision Tree. Our proposal has been evaluated for two contexts with quite different operational features, showing that T-DTC is one of the best algorithm for both network environments. In conclusion, our proposed algorithm definitely enhances the prospects of ensemble learning to be applied to real time NTC.

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. Finally, we would like to thank the ISP for the real network traffic captures and the resources shared with us for this work.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] L. Hernández, C. Baladrón, J. Aguiar, B. Carro, and A. Sánchez-Esguevillas, "Classification and Clustering of Electricity Demand Patterns in Industrial Parks," *Energies*, vol. 5, no. 12, pp. 5215–5228, Dec. 2012.

[6] L. Hernández, C. Baladrón, J. M. Aguiar, B. Carro, A. Sánchez-Esguevillas, and J. Lloret, "Artificial neural networks for short-term load forecasting in microgrids environment," *Energy*, vol. 75, pp. 252–264, Oct. 2014.

[7] 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.

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

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

[10] 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.

[11] 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.

[12] P. Casas, J. Mazel, and P. Owezarski, "MINETRAC : Mining Flows for Unsupervised Analysis & Semi-Supervised Classification."

[13] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 1–34, Mar. 2012.

[14] 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.

[15] 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.

[16] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and Continuous Machine-Learning-Based Classification for Interactive IP Traffic," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1880–1894, Dec. 2012.

[17] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification," *Comput. Networks*, vol. 55, no. 5, pp. 1083–1099, Apr. 2011.

[18] 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.

[19] 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.

[20] 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.

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

[22] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Y. Lee, "Internet traffic classification demystified: myths, caveats,

and the best practices," *Traffic*, vol. 50, no. 4, pp. 1–12, 2008.

[23] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognit.*, vol. 36, pp. 849–851, 2003.

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

[25] 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.

[26] "nDPI." .

[27] 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.

[28] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," *Int. Conf. Mach. Learn.*, pp. 1–8, 2003.

[29] B. Senliol, G. Gulgezen, L. Yu, and Z. Cataltepe, "Fast Correlation Based Filter (FCBF) with a different search strategy," *2008 23rd Int. Symp. Comput. Inf. Sci. Isc. 2008*, 2008.

[30] S. E. Gómez, "FCBF_module," 2016. [Online]. Available: https://github.com/SantiagoEG/FCBF_module.

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

[32] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci. (Ny).*, vol. 180, no. 10, pp. 2044–2064, 2010.

[33] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, no. 1/2, pp. 105–139, 1999.

[34] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, 2000.

[35] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A Comparison of Decision Tree Ensemble Creation Techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, Jan. 2007.

[36] T. G. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Jouranal Artifical Intell. Res.*, vol. 2, pp. 263–286, 1995.

[37] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[38] J. Milgram, M. Cheriet, and R. Sabourin, "'One Against One' or 'One Against All': Which One is Better for Handwriting Recognition with SVMs?," *Tenth Int. Work. Front. Handwrit. Recognit.*, pp. 1–6, 2006.

[39] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[40] T. G. Dietterich, "Ensemble Methods in Machine Learning," 2000, pp. 1–15.

[41] K. C. Lan and J. Heidemann, "A measurement study of correlations of Internet flow characteristics," *Comput. Networks*, vol. 50, no. 1, pp. 46–62, 2006.

[42] P. Carvalho, P. Solis, B. Queiroz, B. Carneiro, and M. Deus, "A Traffic Analysis per Application in real IP/MPLS Service Provider Network," in *2007 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks*, 2007, pp. 1–5.

Annex 1: Collection of Statistical Attributes

Table A1 contains the whole collection of statistical attributes used in this work along with a brief description of each one. Note that many statistics have been computed accounting for only one flow direction or both, “direction 0” denotes the way in which first packets of flow connections were detected and vice versa for “direction 1”. The datasets are publicly available at []. This collection of statistics attributes could be improved adding new and more informative attributes extracted from IP and Transport layers.

Table A1. Collection of statistics employed for the experiments presented in this work

Feature Name	Description
%Packet0	Percentage of packets transferred in the direction 0
%Packet1	Percentage of packets transferred in the direction 1
%Bytes0	Percentage of bytes transferred over number of packets in the direction 0
%Bytes1	Percentage of bytes transferred over number of packets in the direction 1
meanBytes0	Mean of bytes transferred over number of packets in the direction 0
meanBytes1	Mean of bytes transferred over number of packets in the direction 1
meanBytesT	Mean of bytes transferred over number of packets in both directions
varBytes0	Variance of bytes transferred over number of packets in the direction 0
varBytes1	Variance of bytes transferred over number of packets in the direction 1
varBytesT	Variance of bytes transferred over number of packets in both directions
rmsBytes0	Root mean square of bytes transferred over number of packets in the direction 0
rmsBytes1	Root mean square of bytes transferred over number of packets in the direction 1
rmsBytesT	Root mean square of bytes transferred over number of packets in both directions
maxBytes0	Maximum number of bytes transferred in the direction 0
maxBytes1	Maximum number of bytes transferred in the direction 1
maxBytesT	Maximum number of bytes transferred in both directions
minBytes0	Minimum number of bytes transferred in the direction 0
minBytes1	Minimum number of bytes transferred in the direction 1
minBytesT	Minimum number of bytes transferred in both directions
meanInterArrivalTime0	Mean of interarrival time over number of packets in the direction 0
meanInterArrivalTime1	Mean of interarrival time over number of packets in the direction 1
meanInterArrivalTimeT	Mean of interarrival time over number of packets in both directions
varInterArrivalTime0	Variance of interarrival time over number of packets in the direction 0
varInterArrivalTime1	Variance of interarrival time over number of packets in the direction 1
varInterArrivalTimeT	Variance of interarrival time over number of packets in both directions
rmsInterArrivalTime0	Root mean square of interarrival time over number of packets in the direction 0
rmsInterArrivalTime1	Root mean square of interarrival time over number of packets in the direction 1
rmsInterArrivalTimeT	Root mean square of interarrival time over number of packets in both directions
maxInterArrivalTime0	Maximum number of interarrival time in one packet in the direction 0
maxInterArrivalTime1	Maximum number of interarrival time in one packet in the direction 1
maxInterArrivalTimeT	Maximum number of interarrival time in one packet in both directions
minInterArrivalTime0	Minimum number of interarrival time in one packet in the direction 0
minInterArrivalTime1	Minimum number of interarrival time in one packet in the direction 1
minInterArrivalTimeT	Minimum number of interarrival time in one packet in both directions
meanTCPWin0	Mean of TCP window sizes over number of packets in the direction 0
meanTCPWin1	Mean of TCP window sizes over number of packets in the direction 1
meanTCPWinT	Mean of TCP window sizes over number of packets in both directions
varTCPWin0	Variance of TCP window sizes over number of packets in the direction 0
varTCPWin1	Variance of TCP window sizes over number of packets in the direction 1
varTCPWinT	Variance of TCP window sizes over number of packets in both directions
rmsTCPWin0	Root mean square of TCP window sizes over number of packets in the direction 0
rmsTCPWin1	Root mean square of TCP window sizes over number of packets in the direction 1
rmsTCPWinT	Root mean square of TCP window sizes over number of packets in both directions
maxTCPWin0	Maximum number of TCP window in one packet in the direction 0
maxTCPWin1	Maximum number of TCP window sizes in one packet in the direction 1
maxTCPWinT	Maximum number of TCP window sizes in one packet in both directions
minTCPWin0	Minimum number of TCP window sizes in one packet in the direction 0

minTCPWin1	Minimum number of TCP window sizes in one packet in the direction 1
minTCPWinT	Minimum number of TCP window sizes in one packet in both directions
flowDurationT	Flow duration accounting for packets in both directions
flowDuration0	Flow duration accounting for packets in the direction 0
flowDuration1	Flow duration accounting for packets in the direction 1
flowSize	Total flow size in bytes accounting for bytes transferred in both directions
meanBytes/Time0	Mean of bytes transferred over flow duration in the direction 0
meanBytes/Time1	Mean of bytes transferred over flow duration in the direction 1
meanBytes/TimeT	Mean of bytes transferred over flow duration in both directions
varBytes/Time0	Variance of bytes transferred over flow duration in the direction 0
varBytes/Time1	Variance of bytes transferred over flow duration in the direction 1
varBytes/TimeT	Variance of bytes transferred over flow duration in both directions
rmsBytes/Time0	Root mean square of bytes transferred over flow duration in the direction 0
rmsBytes/Time1	Root mean square of bytes transferred over flow duration in the direction 1
rmsBytes/TimeT	Root mean square of bytes transferred over flow duration in both directions
meanTCPWin/Time0	Mean of TCP window sizes over flow duration in the direction 0
meanTCPWin/Time1	Mean of TCP window sizes over flow duration in the direction 1
meanTCPWin/TimeT	Mean of TCP window sizes over flow duration in both directions
varTCPWin/Time0	Variance of TCP window sizes over flow duration in the direction 0
varTCPWin/Time1	Variance of TCP window sizes over flow duration in the direction 1
varTCPWin/TimeT	Variance of TCP window sizes over flow duration in both directions
rmsTCPWin/Time0	Root mean square of TCP window sizes over flow duration in the direction 0
rmsTCPWin/Time1	Root mean square of TCP window sizes over flow duration in the direction 1
rmsTCPWin/TimeT	Root mean square of TCP window sizes over flow duration in both directions
NPKT_64	Number of packets with equal or less than 64 bytes in both directions
NPKT_128	Number of packets with equal or less than 128 bytes in both directions
NPKT_256	Number of packets with equal or less than 256 bytes in both directions
NPKT_512	Number of packets with equal or less than 512 bytes in both directions
NPKT_1024	Number of packets with equal or less than 1024 bytes in both directions
NPKT_MORE	Number of packets with more than 1024 bytes in both directions

A.3 Journal Paper. Exploratory Study on Class Imbalance and Solutions for Network Traffic Classification

Table A3. JCR-Indexed Paper Information	
Title	Exploratory Study on Class Imbalance and Solutions for Network Traffic Classification
Authors	Santiago Egea Gómez, Luis Hernández-Callejo, Belén Carro Martínez and Antonio Sánchez-Esguevillas
Journal	Neurocomputing (IF: 4.072)
Volume	Volume 343, 28 May 2019, Pages 100-119
Publication Date	4 February 2019
DOI	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.

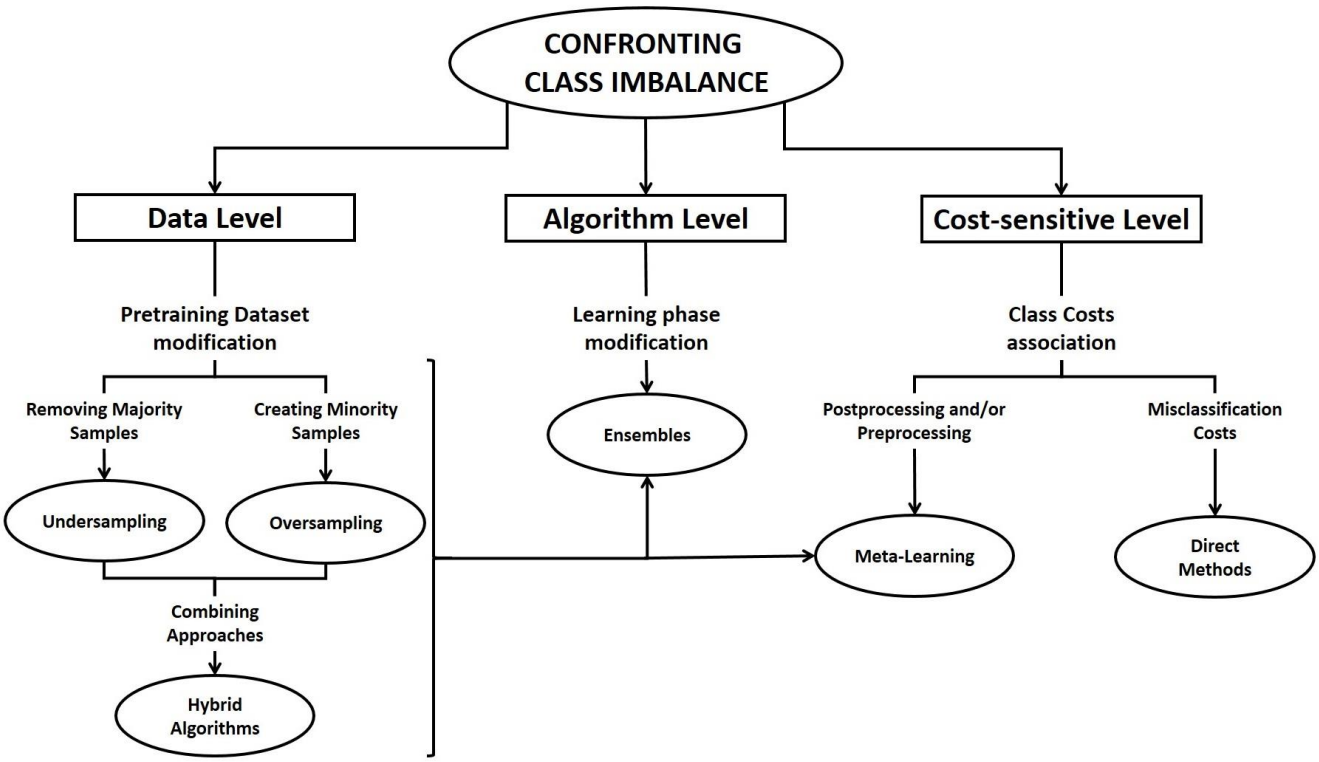

Fig. 1. Categorization of solutions to Class Imbalance

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.

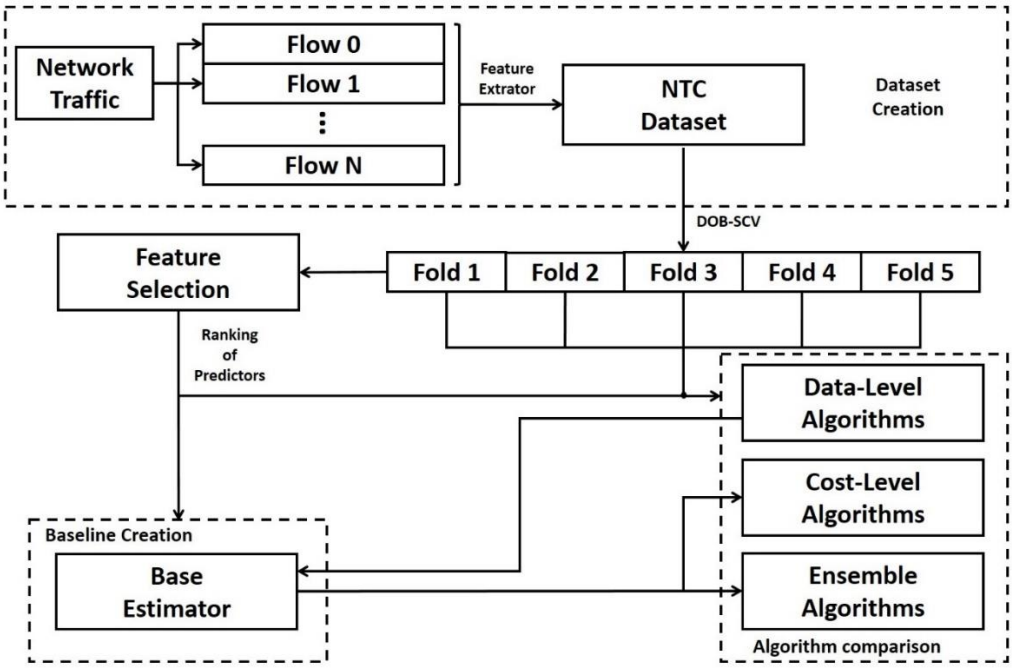

Fig. 2 Methodology Overview

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.

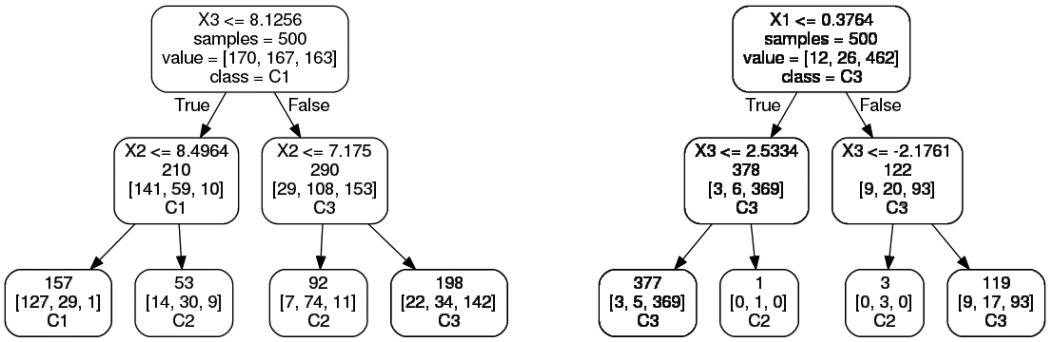

Fig. 3. Internal set of decision rules implemented by Decision Tree. The classes to predict are C1, C2 and C3; and the predictors are X1, X2 and X3

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.

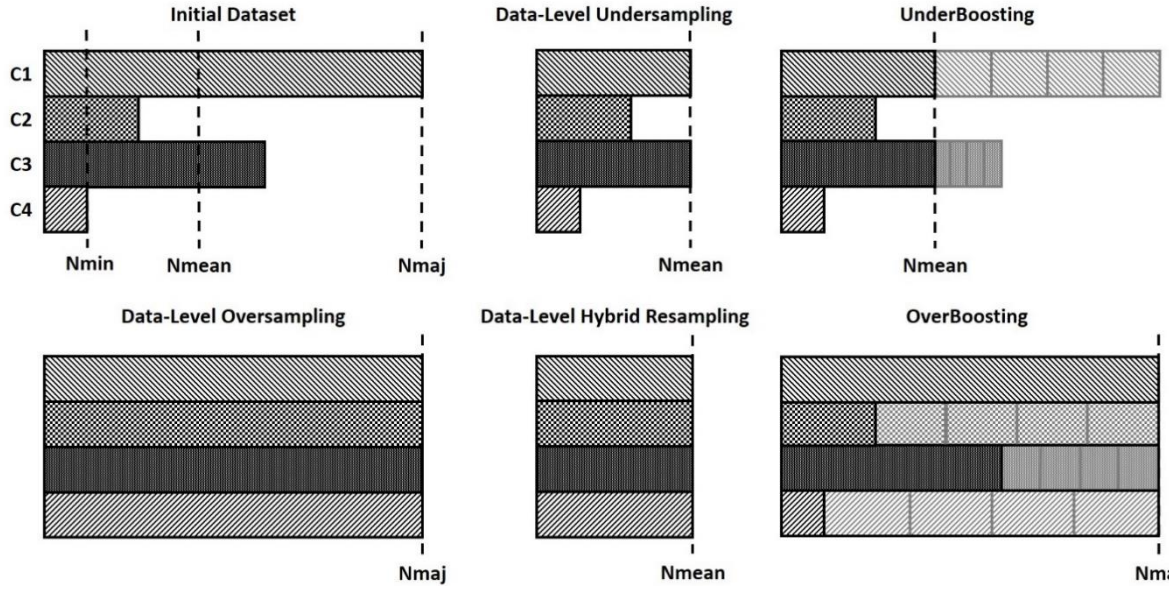

Fig. 4. Strategies to adjust resampling ratios. C1, C2, C3 and C4 denote arbitrary classes, N_{min} the minimum population, N_{maj} the maximum population and N_{mean} the mean of all populations

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].

A.4 Conference Paper. Exploratory Study on Class Imbalance and Solutions for Network Traffic Classification

Table A3. JCR-Indexed Paper Information	
Title	A Feature Selection Framework and a Predictors Study for Internet Traffic Classification
Authors	Santiago Egea Gómez, Luis Hernández-Callejo, Belén Carro Martínez and Antonio Sánchez-Esguevillas
Conference	ASPAI-2019
Conference Date	20-22 March 2019

1st International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2019), 20-22 March 2019, Barcelona, Spain

Oral

Topic: *<please select the mail topic from the ASPAI' 2018 Conference's Call for Paeprs>*

A Feature Selection Framework and a Predictors Study for Internet Traffic Classification

Santiago Egea Gómez[1], Luis Hernández-Callejo[2], Belén Carro Martínez[1] and Antonio Sánchez-Esguevillas

[1] University of Valladolid, Castilla y León, Valladolid, Spain

[2] University of Valladolid, Campus Universitario Duques de Soria, Soria, Spain

Tel.: + 34 983423980

E-mail: santiago.egea@alumnos.uva.es

Summary: Network traffic classification (NTC) has attached attention due to its relevance for traffic control in enabling technologies, taking a significant prominence approaches based on Machine Learning (ML). Being Feature Selection (FS) an essential for ML classification, a new FS framework is presented here for efficient early NTC. Our proposal combines filter and wrapper methods to increase the diversity in attribute selection, these strategies independently produce predictor rankings that are used in a final subset selection. The proposal is tested against datasets extracted from a quite challenging network scenario. The proposal was validated against different NTC datasets extracted from various data sources contained in packet headers. The presented results probe that our FS method is effective in reducing the problem dimensionality preserving or even improving classifier performances. Furthermore, we discuss the predictive power of different sets of predictors for early NTC. Finally, we performed a minor experiment to assess the effect of port evasion when port numbers are included in the predictive model.

Keywords: Machine Learning, Feature Selection, Network Traffic Classification, Supervised Learning, Network managent

1. Introduction

Enabling technologies rely on underlying Internet networks as means of communication. Smart Cities and Internet of Things envision facilities (such as: control of critical infrastructures, assistance in emergencies and smart transportation) requiring traffic control for availability, privacy and security [1]. In this vein, NTC constitutes a key piece to detect service decays and cyber threats.

ML has attached a relevant prominence for NTC, since it enables accurate traffic classification evading the handicaps of previous approaches [2]. An essential in ML is FS. Through FS, the best predictors are selected for training, meanwhile irrelevant ones are discarded leading to efficient classification models. Three FS approaches exist according to their selection schemes: (1) Filter methods, which assume an information metric to assess the quality of predictors; (2) Wrapper methods, which evaluate the importance of attributes using learning algorithms; and (3) Embedded methods, which are integrated in learning algorithms while training. In this paper, an FS framework is proposed for efficient NTC. The predictors are separately ranked by several filter and wrapper strategies, and afterwards the rankings are combined to select the final subset. Furthermore, we analyze several sets of predictors from different raw information contained in packet headers.

This article is structured as follows. Section 2 reviews previous works in FS for NTC. Section 3 describes the methodology followed and presents our FS framework. The experimental results are presented and discussed in Section 4, and finally we draw the conclusions.

2. Previous Works

FS for efficient ML-based NTC is not unexplored, and many authors have proposed their solutions. In [3], A. Fahad et al. presented an FS method called Global Optimization Approach (GOA) to find a stable set of predictors over time. GOA combines a preselection phase based on filters and a subsequent wrapper scheme based on Random Forest. A class-oriented FS algorithm (COFS) and an ensemble classifier were proposed in [4]. First, COFS selects a preliminary subset of attributes for each class, and redundant attributes are removed according to Weighted Symmetric Uncertainty (WSU).

A feature extraction and selection approach were presented in [5]. Feature extraction is based on Wavelet Multifractal transformation on raw packet-header information. As for attribute selection, Principal Component Analysis (PCA) is used to filter out the irrelevant components, and K-Means to cluster the features that are optimal or redundant. M. Shafiq et al. [6] presented a wrapper method to select the best predictors for imbalanced NTC. The proposed method filters out the irrelevant attributes using Weighted Mutual Information metric, and a learning algorithm is used to assess the AUC-ROC for each predictor.

Finally, Efficient Feature Optimization Approach (EFOA) is proposed [7] to confront Class Imbalance and concept of drift in NTC. This proposal includes

feature generation using Deep Belief Networks and a subsequent selection based on WSU.

Our FS framework presents relevant differences respecting previous approaches, since we combine several information metrics and performance metrics to improve the diversity for predictor selection.

3. Methods and Materials

3.1. Network Environment and Datasets

The traffic data employed here was captured in a backbone at an ISP network, which constitutes a challenging scenario. The traffic was sniffed in a high-speed link supporting transmission rates up to 7Gbps, where connections are susceptible to packet losses and multipath effect. We have employed two different datasets for training and validating the NTC models. The tranining data comprises 12Gb collected on 17/1/2017 for 5 minutes, meanwhile the validation consists of 35.62GB captured for almost 10 minutes on 23/3/2017.

The first five packet-headers were processed for each flow to create the classification objects and following the concept of early NTC [8]. There are mainly four information sources available in IP and transport layers: packet sizes, timestamps, window sizes and others parameters (duration, directions, ports and so on). For the three first of them, a set of 47 predictors were computed and merged with the original raw information, as Fig. 1 illustrates. The collection of predictors includes statistics (means, maximum and minimum values, …) and FFT transform components. In the case of "Others" predictors, we considered ports, directions, % of packets in each direction and packet counts. Finally, we merged all those datasets resulting in a whole dataset with a total of 172 attributes. The different datasets are denoted as follows: SIZES, IATS, WINSIZES, OTHERS and WHOLE.

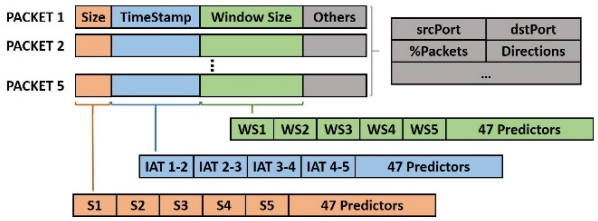

Fig. 1. Collection of attributes and datasets employed for our experiments

3.2. FS Framework based on rankings

Fig. 2 presents the proposed FS framework. As it is shown, three phases are involved: (1) Filter, (2) Wrapper, and (3) Final Ranking. During (1), several independent attribute rankings are computed using different filters, meanwhile wrapper strategies rank predictors in (2). The more relevant the attribute, the topper the position it occupies in the rankings, so that the attribute positions are used as scores and combined in means to select the final subset in (3).

Through combining filters, the attributes are assessed against different information-based metrics, therefore increasing the diversity in the selection. In (2), several learning algorithms and performance metrics are assumed leading to more rich selection criteria. This framework follows a flexible design that allow extending it including more selection strategies.

For our experiment, we selected the following nine filter methods using different information metrics: MRMR [9], CIFE [10], CMIN [11], ICAP [12], MIFS [13], DISR [14], JMI [15] and MIM [16]. The most of the filters selected are implemented in the Python library, with the exception of FCBFiP, which was used in [17] and is available in [18]. Regarding Wrapper Ranking, we have evaluated three performance metrics for three learning algorithms. As Decision Tree algorithms have shown as a promising approach for NTC due to its excellent ratio amongst precision and speed, we have selected the CART Decision Tree and two ensemble algorithms implemented in Scikit-learn [19]. The OutputCode and Bagging algorithms were the ensemble techniques considered in wrapper ranking. In (2), our FS framework ranks the attributes according to the performance metrics they produce when they are used as unique predictor for each learning algorithm. The selected performance metrics in this phase are: Overall Accuracy (OA), Byte Accuracy (BA) and geometric mean (GM).

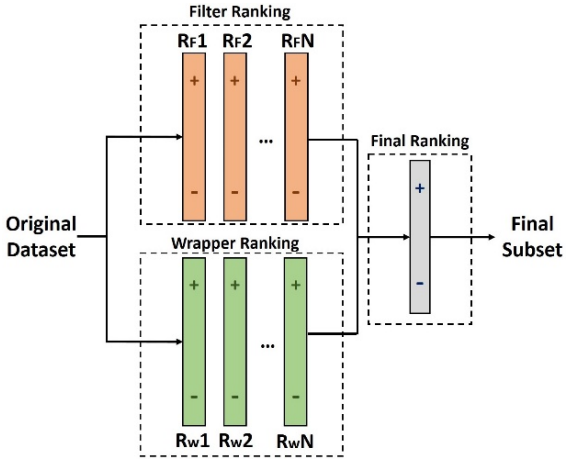

Fig. 2. FS Framework based on rankings.

4. Experimental results

In this section, we present and discuss the results obtained from applying our FS framework to the different datasets. Firstly, we present a preliminary experiment to assess predictive power of the different sets of predictors (SIZES, IATS, WINSIZES and OTHERS) and confirm the efficiency of our selection approach. In a second experiment, we applied our FS framework to the WHOLE dataset to assess which are the most relevant attributes when several information sources are combined. Furthermore, we performed a minor experiment to evaluate how port evasion may affect to performance metrics when port numbers are included as attributes in traffic classifiers.

4.1. Preliminary Results

Through this experiment, we pretend to validate our FS framework and study which family of predictors are the most relevant for early NTC. The final ranking for each dataset is employed to sequentially vary the number of attributes selected, and a Decision Tree is trained to evaluate the model performances according to OA, BA and GM. Figure 3 presents the results obtained for the different models, and the top ten attributes are shown in Table 2 for each case.

From Figure 3, we find that the best-performing attributes in terms of OA are OTHERS, with which we were able to identify more than 82% of connections. Conversely, the WINSIZES and SIZES obtained OAs around 58% and 51% respectively. While on BA, SIZES clearly outperformed the other sets of predictors accomplishing BAs up to 94.7%. The rest of datasets did not produced as positive BAs as SIZES, and the second best BAs were obtained by WINSIZES with values around 80%. Focusing on GM, we observe that OTHERS is anew the set of attributes producing the best results reaching GMs greater than 65% for certain subset sizes. In the case of SIZES, the GM overcame 23% for 7 and 8 predictors; on contrast to IATs and WINSIZES datasets that yielded null GMs. OTHERS and IATS produced quite weak results for this performance metric. Note also that the subsets computed by our FS framework accomplished similar performances to that using all predictors for each datasets, which confirms the effectiveness of our proposal in reducing the dimensinality space without performance decays.

If we observe Table 2, we find that only there is one FFT component (8th) amongst the best ten features for SIZES. On the contrary, raw packet sizes (2nd, 7th and 9th) and statistical components (such as: means root mean square and maximum values) have a notable presence in the selected subset (1st, 3rd, 4th, 5th, 6th and 10th). While on IATS, FFT-related predictors (such as module and phase of FFT components) were selected (1st, 6th, 7th, 9th and 10th) jointly with an important number of statistics (2nd, 3rd, 4th and 5th), meanwhile only one raw attribute was selected (8th). Focusing on WINSIZES, we observe that two raw predictors (2nd and 8th) and two FFT phases (4th and 10th) were selected for the final subset, in contrast to statistical attributes that mainly composed the top ten subset (1st, 3rd, 5th, 6th, 7th and 9th). Unlike the previous datasets, OTHERS dataset contains predictors of other nature. Interestingly, we find that source and destination ports (1st and 2nd) have an important impact on the final subset resulting from our FS framework. Furthermore, we find that packet-size counts exhibited notable predictive power for NTC according to their positions in the ranking (3rd, 4th, 5th, 6th and 10th). Conversely, packet directions (6th and 8th) and percentage of exchanged packets (7th and 9th) were not as important as the former attributes.

Generally, the poor performances exhibited by IATS for all metrics considered might be caused by the instability of this family of predictors. IATS are quite susceptible to the operation phase of the Internet network. When the workload is very high packet forward slows down, which produces variations in these types of attribute leading to performance losses.

In the case of WINSIZES, something similar might happen. TCP window sizes are variable parameters that protocols adjust depending on the network workload, so that attributes related to this parameter could vary on time. Another important handicap of these kinds of attribute is that window sizes are not useful for UDP connections.

As its high GM reveals, OTHERS attributes provided predictiveness to identify different types of application. However, port numbers had an important role for this collection of predictors. As port numbers are configurable parameters, models including these parameters are quite susceptible to port evasion, which is a severe handicap.

In the case of SIZES, these attributes are independent from the transport protocol. Similarly to OTHERS, the GM obtained for SIZES indicates that this type of attributes are useful to identify different family of applications, in contrast to the rest of datasets that obtained almost null GMs.

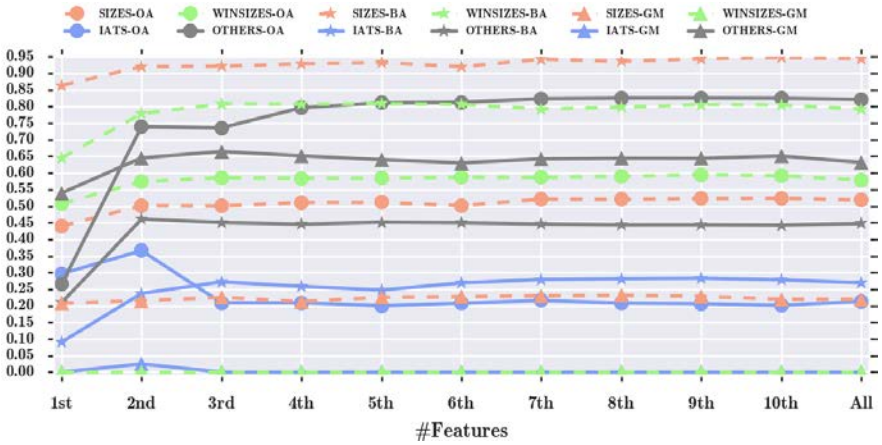

Fig. 3. Performances for different subet sizes

Table 1. Top ten predictors for the different datasets

		Top ten predictors
SIZES		
	1st	Root Mean Square of packet sizes in both directions.
	2nd	Size of the 1st packet exchanged.
	3rd	Maximum packet size in the 1st direction.
	4th	Root Mean Square of packet sizes in the 1st direction.
	5th	Maximum packet size in both directions.
	6th	Mean of packet sizes in the 1st direction.
	7th	Size of the 2nd packet exchanged.
	8th	Phase of the 1st FFT component on sizes in the 1st direction
	9th	Size of the 5th packet exchanged.
	10th	Mean of packet sizes in both directions.
IATS		
	1st	Phase of the 1st FFT component on IATs in both directions.
	2nd	Percentage of IATs in the 1st direction.
	3rd	Maximum IAT in both directions.
	4th	Root Mean Square of IATs in both directions.
	5th	Root Mean Square of IATs in the 1st direction.
	6th	Module of the 5th FFT component on IATs in both directions.
	7th	Phase of the 1st FFT component on IATs in both directions.
	8th	IAT of the 2nd packet.
	9th	Module of the 3rd FFT component on IATs in the 1st direction.
	10th	Module of the 2nd FFT component on IATs in the 1st direction.
WINSIZES		
	1st	Root Mean Square of window sizes in both directions.
	2nd	Window size of the 1st packet exchanged.
	3rd	Mean of window sizes in the 1st direction.
	4th	Phase of the 1st FFT component on Window sizes in the 1st direction.
	5th	Maximum window size in the 1st direction.
	6th	Maximum window size in both directions.
	7th	Root Mean Square of window sizes in the 1st direction.
	8th	Window size of the 2nd packet exchanged.
	9th	Percentage of window sizes in the 1st direction
	10th	Phase of the 3rd FFT component on Window sizes in the 1st direction.
OTHERS		
	1st	Destination port number.
	2nd	Source port number.
	3rd	Number of packets with packet sizes between 128 and 64 bytes.
	4th	Number of packets with packet sizes between 10 and 20 kilobytes.
	5th	Number of packets with packet sizes greater than 64 bytes.
	6th	Direction of the 5th packet.
	7th	Percentage of packets in the 1st direction.
	8th	Direction of the 2nd packet.
	9th	Percentage of packets in 1st direction.
	10th	Number of packets with packet sizes greater than 20 kilobytes.

4.2. Final subset

In this section we present and discuss the results obtained from applying our FS framework to the WHOLE dataset. Table 3 contains the top ten predictors selected by our FS method, and the performance metrics obtained when a Decision Tree is trained including each one.

Observing Table 3, we find that the five top predictors yielded better outcomes that the original dataset according to OA and GM, on contrast to BA that was slightly lower than using all attributes. The highest OA was reached when the 9th predictor was included in the predictive model identifying accurately the 98.5% of connection flows. In the instance of BA, the best value was obtained when eight attributes are selected, although subsets achieved the same BA as the whole dataset when more than five predictors were selected. While on GM, the best performance was achieved when six or seven attributes are selected for training. GM accomplished 84% for these subset sizes, meanwhile the whole datasets got a score of 73.5%. The notable increase in this performance metric reveals that a smaller dataset better identifies a range of diverse applications, providing better outcomes in presence of Class Imbalance.

Table 2. The ten best-scored predictors and performance metrics for the WHOLE dataset

Top ten predictors		OA	BA	GM
1st	Destination port number.	.265	.209	.539
2nd	Size of the 1st packet exchanged.	.829	.848	.607
3rd	Source port number.	.961	.969	.697
4th	Maximum packet size in the 1st direction.	.956	.989	.720
5th	Window size of the 1st packet exchanged.	.983	.995	.824
6th	Root Mean Square of packet sizes in both directions.	.981	.996	**.840**
7th	Phase of the 1st FFT component on packet sizes in the 1st direction.	.981	.996	**.840**
8th	Maximum packet size in both directions.	.980	**.997**	.817
9th	Phase of the 1st FFT component on Window sizes in the 1st direction.	**.985**	.996	.820
10th	Maximum window size in both directions.	.984	.996	.811
Whole dataset		.979	.996	.735

Through this experiment, we have probed that our FS framework is able to reduce the training subsets increasing some performance metrics for early NTC. We also found that the combination of predictors computed using different parameters from packet headers yielded much better results than independently using them. Note that the collection of top ten predictors (Table 3) includes source and destination port numbers (1st and 3rd). As aforementioned, these parameters are susceptible to evasion, since port numbers are a configurable parameter by users or applications. Below, we present the results of an experiment during which port evasion was simulated to assess performance losses due to this effect. In addition to port numbers, the top ten subset contains five predictors computed from packet sizes (2nd, 4th, 6th, 7th and 8th) and three Window-size related attributes (5th, 9th and 10th).

Masking connections behind random port numbers

Finally, we provide a minor experiment to probe that ML-based traffic classifiers are susceptible to port evasion when port numbers are included in the predictive model. For this purpose, we have selected the subset with six features from Table 3, and port evasion was simulated by randomly modifying the source and destination port numbers for specific percentage of samples. Fig. 4 contains the results obtained in this experiment.

Generally, all performance metrics are negatively affected when the connections are masked behind other ports. When the percentage of masked connections increases, the performances of the classifier notably decreases. In the case of masking the 5%, all metrics decreased in around 4%. Through this simple experiments, we have probed as port evasion can decrease the performance of ML-based traffic classifiers when port numbers are included in the predictive model.

5. Conclusions

Through this work, we have presented an FS framework for selecting a reduced dataset for early NTC. Our framework combines two parallel ranking phases in which filter methods and wrapper strategies are employed to independently rank predictors according to their relevance in the predictive model. We have validated our FS scheme on different NTC datasets containing predictors computed from the different parameters in packet headers (such as: packet sizes, inter-arrival times, window sizes and so on).

As a result of our experiments, we have found that our FS framework is able to notably reduce the attribute space preserving and even improving the performances of the classifier. Additionally, we have analyzed the different collections of attributes to find out the predictive power of each one for NTC. Finally, we have applied our FS framework to a whole dataset that combines the former datasets. As result, we found that combining predictors computed from different network parameters provides better results than employing them separately. As source and destination port numbers were ranked as one of the most relevant attributes, we performed a minor experiment to assess the effect of port evasion on classifier performances. Our results reveal that port evasion decreases the classifier accuracy, and it should be considered when port numbers are included in the predictive model.

Acknowledgements

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. Finally, we would like to thank the ISP for the real network traffic captures and the resources shared with us for this work.

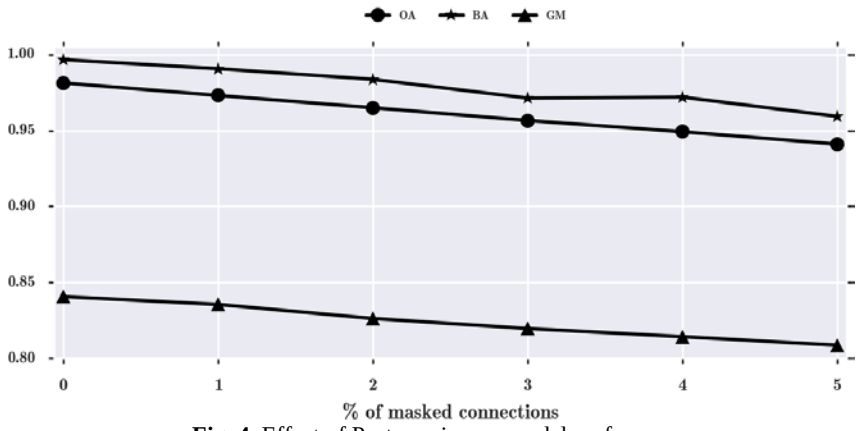

Fig. 4. Effect of Port evasion on model performances

References

[1] T. K. L. Hui, R. S. Sherratt, and D. Díaz, "Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies," *Futur. Gener. Comput. Syst.*, vol. 76, pp. 358–369, 2017.

[2] 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.

[3] A. Fahad, Z. Tari, I. Khalil, A. Almalawi, and A. Y. Zomaya, "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion," *Futur. Gener. Comput. Syst.*, vol. 36, pp. 156–169, 2014.

[4] 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.

[5] 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.

[6] M. Shafiq, X. Yu, A. Kashif, B. Hassan, N. Chaudhry, and D. Wang, "A machine learning approach for feature selection traffic classification using security analysis," *J. Supercomput.*, 2018.

[7] H. Shi, H. Li, D. Zhang, C. Cheng, and X. Cao, "An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification," vol. 132, pp. 81–98, 2018.

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

[9] Hanchuan Peng, Fuhui Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[10] D. Lin and X. Tang, "Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion," Springer, Berlin, Heidelberg, 2006, pp. 68–82.

[11] F. Fleuret, "Fast Binary Feature Selection with Conditional Mutual Information," *J. Mach. Learn. Res.*, vol. 5, pp. 1531–1555, 2004.

[12] A. Jakulin, "Machine Learning Based on Attribute Interactions," *Thesis*, pp. 1–252, 2005.

[13] R. Battiti, "Using Mutual Information for Selecting Features in Supervised Neural-Net Learning," *Ieee Trans. Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[14] P. E. Meyer, C. Schretter, and G. Bontempi, "Information-Theoretic Feature Selection in Microarray Data Using Variable Complementarity," *IEEE J. Sel. Top. Signal Process.*, vol. 2, no. 3, pp. 261–274, 2008.

[15] H. H. Yang and J. Moody, "Data Visualization and Feature Selection: New Algorithms for Nongaussian Data," *Adv. Neural Inf. Process. Syst.*, vol. 12, no. Mi, pp. 687–693, 1999.

[16] D. D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," pp. 212–217, 1992.

[17] 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.

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