Abstract—Tor is an anonymous Internet communication system based on the second generation of onion routing network protocol. Using Tor is really difficult to trace the users Internet activity: this is the reason why the usage of Tor is intended in order to protect the privacy of users, their freedom and the ability to conduct confidential communications without being monitored. Tor is even more used by cyber-criminals in order to cover their illegal activities: the Tor community has observed, for instance an alarming increase in the number of malware that abuse of the popular anonymizing network to hide their command and control infrastructures. In this paper we present a technique able to identify whether an host is generating Tor-related traffic. We resort to well-known machine learning algorithms in order to evaluate the effectiveness of the proposed feature set in a real world environment. In addition we demonstrate that the proposed method is able to recognize the kind of activity (e.g., email or P2P applications) the user under analysis is doing on the Tor network.

I. INTRODUCTION AND BACKGROUND

The Tor network (i.e., the Onion Router network) enables the users in order to surf the Internet, chat and send instant messages anonymously.

Basically the Tor network is a group of volunteer-operated servers that allows people to improve their privacy and security on the Internet. The Tor users employ this network by connecting through a series of virtual tunnels rather than making a direct connection, thus allowing both organizations and individuals to share information over public networks without compromising their privacy [1]. Along the same line, Tor is an effective censorship circumvention tool, allowing its users to reach otherwise blocked destinations or content. Tor can also be used as a building block for software developers to create new communication tools with built-in privacy features.

Onion routing is implemented by encryption in the application layer of a communication protocol stack, nested like the layers of an onion. Tor encrypts the data, including the next node destination IP address, multiple times and sends it through a virtual circuit comprising successive, randomly selected Tor relays [2]. Each relay decrypts a layer of encryption to reveal only the next relay in the circuit in order to pass the remaining encrypted data on to it. The final relay decrypts the innermost layer of encryption and sends the original data to its destination without revealing, or even knowing, the source IP address [3]. Because the routing of the communication is partly concealed at every hop in the Tor circuit, this method eliminates any single point at which the communicating peers can be determined through network surveillance that relies upon knowing its source and destination.

Tor is not meant to completely solve the issue of anonymity on the web. Tor is not designed in order to completely erase tracks but instead to reduce the likelihood for sites to trace actions and data back to the user [1].

Tor is increasingly used for non legal activities i.e., to gain access to censored information, to organize political activities [4], or to circumvent laws against criticism of heads of state. Tor has, for instance, been used by criminal enterprises, hacktivist groups, and law enforcement agencies at cross purposes, sometimes simultaneously; likewise, agencies within the U.S. government variously fund Tor [5].

Tor has been described by The Economist, in relation to Bitcoin and Silk Road, as being ”a dark corner of the web” [6].

Starting from these considerations, in this paper we focus on the analysis and identification related to Tor network traffic. Considering a real-world traffic flow, our aim is to identify whether the traffic flow under analysis is generated by the Tor network or not. In addition to Tor network traffic identification, we focus also on the traffic characterization i.e., on the kind of use that the user is doing on the basis of seven different categories we defined: browsing, email, chat, streaming, FTP, VoIP and P2P.

In last years, researchers proposed several solutions in order to identify Tor network generated traffic. For instance, Lashkari et al. [7] consider a time analysis on Tor traffic flows, captured between the client and the entry node. The obtain 0.95% using the RandomForest classification algorithms.

Authors in [8] the authors propose a method based on HMM (Hidden Markov Models) to classify Tor traffic in 4 categories: P2P, FTP, IM and Web. As features they use burst volumes and directions, extracted from Tor flows. They use HMM to build ingress and egress models of the different application types (P2P, FTP, IM and Web). They obtain an overall accuracy value of 92

2http://www.csoonline.com/article/2131589/investigations-forensics/how-online-black-markets-work.html
Chakravarty et al. [9] present an attack against the Tor network, with the objective of revealing the IP address of the clients. They discuss an active traffic analysis attack based on user traffic at the server side and they observe a similar perturbation at the client side through statistical correlation. Their method achieves an accuracy of 100% in in-lab tests, and more than 81% in real-world experiments.

Basically the paper poses following research questions:

• RQ1: is it possible to discriminate whether the user network traffic is generated using the TOR network using network-based features?
• RQ2: is it possible to identify the used services when the user is using the Tor network?

Similarly to the discussed methods, we evaluate the effectiveness of state-of-the-art machine learning algorithms in order to detect whether the traffic flow is generating through Tor network. Indeed, under a general umbrella, exploiting machine learning approaches to network analysis and detection is a major trend of recent years (e.g., [10], [11]). As secondary point we identify the kind of activity the user is performing an the basis of the seven categories we defined. As demonstrated by the evaluation, the proposed method is able to obtain a very high accuracy.

The reminder of the paper is organized as follows: Section II introduces the method, Section III illustrates the results of the experiment. Finally, conclusions and future works are given in Section IV.

II. The Method

In this section we present the feature set we propose in order to discriminate between Tor network traffic and normal one.

A. The Proposed Features

In order to explain the proposed feature set we consider the common definition of flow, where a flow is defined by a sequence of packets with the same values: {Source IP, Destination IP, Source Port, Destination Port and Protocol (TCP or UDP)}. In the case of Tor traffic, all flows will be TCP, since it does not support UDP. Along with the flow generation the considered features are associated with each flow.

To evaluate the effectiveness of the method we propose, we consider a real-world dataset with network traffic generated by the Tor network extracted. The traffic was captured using the Wireshark and tcpdump tools, generating a total of 22GB of data. To facilitate the labeling process the outgoing traffic at the workstation and the gateway is captured simultaneously, collecting a set of pairs of .pcap files: one regular traffic pcap (workstation) and one Tor traffic pcap (gateway) file. The output of this step is a set of labeled traces related to the normal and Tor traffic. Afterward, the packet gathering analysis, the ISCFlowMeter application [12] is considered in order to generate the flows and calculate all the parameters. The FlowMeter generates bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence the statistical time-related features are also calculated separately in the forward and reverse direction. The TCP flows are usually terminated upon connection tear-down (by FIN packet) while UDP flows are terminated by a flow timeout.

We considered the following 23 features:

• fiat: Forward Inter Arrival Time , the time between two packets sent forward direction (mean, min, max, std).
• biat: Backward Inter Arrival Time, the time between two packets sent backwards (mean, min, max, std).
• flowiat: Flow Inter Arrival Time, the time between two packets sent in either direction (mean, min, max, std).
• active: The amount of time a flow was active before going idle (mean, min, max, std).
• idle: The amount of time a flow was idle before becoming active (mean, min, max, std).
• fb psec: Flow Bytes per second.
• fp psec: Flow packets per second.
• duration: The duration of the flow. With exception for the duration, which shows the total time of one flow, there are six groups of features. The first three groups are namely: -fiat, -biat, and -flowiat, and are focused respectively on the forward, backward and bi-directional flows. The fourth and fifth groups of features, are calculated regarding to the idle-to-active or active-to-idle states and are named -idle and -active. Finally, the last group focuses on the size and number of packets per second and is named -psec feature.

B. Evaluation: Study design

We designed an experiment in order to evaluate the effectiveness of the proposed technique, expressed through the research questions stated in the introduction.

More specifically, the experiment is aimed at verifying whether the feature set is able to classify the Tor network traffic by the normal one. The classification is carried out by using a classifier built with the feature set discussed in the previous section.

The evaluation we performed consists of two different stages: (i) the hypotheses testing, to verify if the traffic-related feature present different distributions for the populations of Tor network and the normal one; and (ii) the classification analysis aimed at assessing whether the considered feature categories are able to correctly discriminate between Tor network traffic flow and normal one.

With regards to the hypotheses testing, the null hypothesis to be tested is:

\[ H_0: \text{There are no statistically significant differences} \]

between the considered features of Tor generated traffic flow and normal one

The null hypothesis was tested with Mann-Whitney (with the p-level fixed to 0.05) and with Kolmogorov-Smirnov Test (with the p-level fixed to 0.05). We chose to run two different tests in order to enforce the conclusion validity. The purpose of these tests is to determine the level of significance, i.e., the
risk (the probability) that erroneous conclusions be drawn: we set in following study the significance level equal to .05, i.e. we accept to make mistakes 5 times out of 100.

The second step of the evaluation is represented by the classification analysis, able to to assess if the features are able to correctly discriminate between Tor generated traffic and normal one.

For the classification task, we resort to the supervised learning approach that is basically composed by two different steps:

1) **Learning Step**: starting from the labeled dataset (i.e., where each feature is related to a class. In our case, the class is represented by the kind of network considered i.e.m TOR or NonTOR), we filter the data in order to obtain a feature vector. The feature vectors belonging to all the users involved in the experiment with the associated labels represent the input for the Machine learning algorithm that is able to build a model from the data it analyzed. The output of this step is the model obtained by the labeled dataset.

2) **Prediction Step**: the output of this step is the classification of a feature vector as belonging to the TOR network or not. Using the model built in the previous phase, we input this model using a feature vector without the label: the classifier will output with their label prediction (i.e., TOR or nonTOR).

For training the classifier, we defined \( T \) as a set of labeled traces \((M, l)\), where each \( M \) is associated to a label \( l \in \{T, N\} \) (where \( T \) represents the Tor trace, while \( N \) the nonTor one). For each \( M \) we built a feature vector \( F \in R_y \), where \( y \) is the number of the features used in training phase (\( y = 23 \)).

For the learning phase, we use a \( k \)-fold cross-validation: the dataset is randomly partitioned into \( k \) subsets. A single subset is retained as the validation dataset for testing the model, while the remaining \( k-1 \) subsets of the original dataset are used as training data. We repeated the process for \( k = 10 \) times; each one of the \( k \) subsets has been used once as the validation dataset. To obtain a single estimate, we computed the average of the \( k \) results from the folds.

We evaluated the effectiveness of the classification method with the following procedure:

1) build a training set \( T \subset D \);
2) build a testing set \( T' = D \setminus T \);
3) run the training phase on \( T \);
4) apply the learned classifier to each element of \( T' \).

Each classification was performed using 90% of the dataset as training dataset and 10% as testing dataset employing the full features set.

The same procedure was performed in order to answer to RQ2: the only differences is represented by the labels that in the second experiments are represented by the different considered categories.

We considered the following state-of-the-art algorithms:

- **J48 Consolided** [16]: it is another J48 that employs the Consolidated Tree Construction algorithm; a single tree is built based on a set of subsamples;
- **BayesNet** [17], [18], [19]: it represents a Bayesian network directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph;
- **jRip** [20], [21], [22]: this algorithm implements a propositional rule learner i.e., the Repeated Incremental Pruning to Produce Error Reduction (RIPPER);
- **OneR** [23], [24]: this algorithm is able to learn a one-level decision tree, i.e. generates a set of rules that test one particular attribute;
- **REPTree** [25], [26], [27]: this algorithm represents the Fast decision tree learner. It basically builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (using backfitting).

The classification analysis was performed using the Weka tool\(^3\), a well-known suite of machine learning software.

The machine used to run the experiments and to take measurements was an Intel Core i5 desktop with 4 gigabyte RAM, equipped with Linux Mint (64 bit), a GNU/Linux desktop distribution.

### III. THE EVALUATION

In order to evaluate the effectiveness of the proposed method we considered a labeled Tor traffic dataset. It was generated using accounts for users Alice and Bob in order to use services like Skype, Facebook, etc. The dataset contains 8 types of traffic (browsing, chat, audio-streaming, video-streaming, mail, VOIP, P2P and File Transfer) from more than 18 representative applications (e.g., Facebook, Skype, Spotify, Gmail etc.). Whonix (https://www.whonix.org), a ready-to-use Linux OS configured to route all traffic through the Tor network has been considered. The Whonix distribution is composed of two virtual machines, the gateway and the workstation. The workstation connects to the Internet through the gateway virtual machine, which in turn routes all the traffic through the Tor network. With this configuration, using the Tor network at the workstation virtual machine becomes transparent. The considered dataset contains the outgoing traffic captured at the workstation and the gateway simultaneously, collecting a set of pairs of .pcap files: one regular traffic pcap (workstation) and one Tor traffic pcap (gateway) file. The considered dataset was labeled in two steps. First, the .pcap files captured at the workstation was processed by extracting the flows in order to confirm that the majority of traffic flows were generated by application X (i.e., Skype, FTPS) with the consequently trace labeling.

\(^3\)http://www.cs.waikato.ac.nz/ml/weka/
We explain in details the eight traffic categories considered in order to recognize the kind of activity the user is doing on the Tor network:

- **Browsing**: Under this label we have HTTP and HTTPS traffic generated by users while browsing (Firefox and Chrome).
- **Email**: Traffic samples generated using a Thunderbird client, and Alice and Bob GMail accounts. The clients were configured to deliver mail through SMTP/S, and receive it using POP3/SSL in one client and IMAP/SSL in the other.
- **Chat**: The chat label identifies instant-messaging applications. Under this label we have Facebook and Hangouts via web browser, Skype, and IAM and ICQ using an application called pidgin.
- **Audio-Streaming**: The streaming label identifies audio applications that require a continuous and steady stream of data. We captured traffic from Spotify.
- **Video-Streaming**: The streaming label identifies video applications that require a continuous and steady stream of data. We captured traffic from YouTube (HTML5 and flash versions) and Vimeo services using Chrome and Firefox.
- **FTP**: This label identifies traffic applications whose main purpose is to send or receive files and documents. For our dataset we captured Skype file transfers, FTP over SSH (SFTP) and FTP over SSL (FTPS) traffic sessions.
- **VoIP**: The Voice over IP label groups all traffic generated by voice applications. Within this label we captured voice-calls using Facebook, Hangouts and Skype.
- **P2P**: This label is used to identify file-sharing protocols like BitTorrent. To generate this traffic we downloaded different .torrent files from the Kali Linux distribution and captured traffic sessions using the Vuze application. We also used different combinations of upload and download speeds.
- **File Transfer**: Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla.

The dataset is available for research purpose.

In order to evaluate the proposed solution as Tor-traffic identifier, four metrics were used to evaluate the classification results: TP Rate, FP Rate, Precision, Recall, F-Measure, MCC, ROC Area and PRC Area.

The precision has been computed as the proportion of examples that were assigned to class X, among all the examples that truly belong to the class, i.e., how much part of the class was captured. It is the ratio of the number of relevant records retrieved to the total number of relevant records:

\[
\text{Precision} = \frac{tp}{tp+fp}
\]

where \(tp\) indicates the number of true positives and \(fp\) indicates the number of false positives.

The recall has been computed as the proportion of examples that were assigned to class X, among all the examples that truly belong to the class, i.e., how much part of the class was captured. It is the ratio of the number of relevant records retrieved to the total number of relevant records:

\[
\text{Recall} = \frac{tp}{tp+fn}
\]

where \(tp\) indicates the number of true positives and \(fn\) indicates the number of false negatives.

The F-Measure is a measure of a test’s accuracy. This score can be interpreted as a weighted average of the precision and recall:

\[
\text{F-Measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

MCC takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes:

\[
\text{MCC} = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}
\]

where \(tn\) is the number of true negatives.

The ROC Area is defined as the probability that a positive instance randomly chosen is classified above a negative randomly chosen.

The PRC Area shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

With regards to the hypotheses testing, we evaluate the null hypothesis and we obtain that all the considered features pass the tests: this is symptomatic that the considered feature set can be a good candidate in order to discriminate between Tor and nonTor network traffic.

Table I shows the obtained results related to RQ1. All the considered algorithms are able to obtain very high performance in Tor traffic identification.

With particular regards to the jRip algorithm, we highlight that this algorithm is able to obtain 1 as precision and recall. Relating to the others algorithms involved in the study, they are able to reach a precision ranging between 0.982 (BayesNet) and 0.998 (J48, J48Consolidated and OneR) and a recall ranging between 0.982 (BayesNet) and 0.998 (J48, J48Consolidated and OneR).

**RQ1 Response**: The considered features are able to discriminate between Tor and nonTor traffic with a precision equal to 1 and a recall equal to 1 when the classification is performed using the jRip algorithm.

Table II shows the obtained results related to RQ2. Relating to the precision and the recall obtained we highlight that:
TABLE I: RQ1 Performance Evaluation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCC</th>
<th>ROC Area</th>
<th>PRC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>0.999</td>
<td>0.007</td>
<td>0.998</td>
<td>0.999</td>
<td>0.999</td>
<td>0.994</td>
<td>0.999</td>
<td>1.000</td>
<td>nonTOR</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.993</td>
<td>0.001</td>
<td>0.997</td>
<td>0.993</td>
<td>0.995</td>
<td>0.994</td>
<td>0.999</td>
<td>0.997</td>
<td>TOR</td>
</tr>
<tr>
<td>J48Consolidated</td>
<td>0.998</td>
<td>0.006</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>0.994</td>
<td>0.999</td>
<td>1.000</td>
<td>nonTOR</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.998</td>
<td>0.002</td>
<td>0.991</td>
<td>0.998</td>
<td>0.994</td>
<td>0.993</td>
<td>1.000</td>
<td>1.000</td>
<td>TOR</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.982</td>
<td>0.020</td>
<td>0.996</td>
<td>0.982</td>
<td>0.989</td>
<td>0.938</td>
<td>0.999</td>
<td>1.000</td>
<td>nonTOR</td>
</tr>
<tr>
<td></td>
<td>0.980</td>
<td>0.018</td>
<td>0.918</td>
<td>0.980</td>
<td>0.948</td>
<td>0.938</td>
<td>0.999</td>
<td>0.995</td>
<td>TOR</td>
</tr>
<tr>
<td>OneR</td>
<td>1.000</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>TOR</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>1.000</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
<td>1.000</td>
<td>1.000</td>
<td>TOR</td>
</tr>
<tr>
<td>J48</td>
<td>0.997</td>
<td>0.000</td>
<td>1.000</td>
<td>0.997</td>
<td>0.999</td>
<td>0.992</td>
<td>0.999</td>
<td>1.000</td>
<td>nonTOR</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.998</td>
<td>0.002</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>0.992</td>
<td>0.999</td>
<td>0.997</td>
<td>TOR</td>
</tr>
<tr>
<td>RepTREE</td>
<td>0.987</td>
<td>0.032</td>
<td>0.997</td>
<td>0.987</td>
<td>0.992</td>
<td>0.940</td>
<td>0.999</td>
<td>0.998</td>
<td>nonTOR</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.982</td>
<td>0.019</td>
<td>0.923</td>
<td>0.983</td>
<td>0.951</td>
<td>0.941</td>
<td>0.998</td>
<td>0.997</td>
<td>TOR</td>
</tr>
</tbody>
</table>

- the J48 algorithm obtains a weight precision and a weight recall equal to 0.998;
- the J48Consolidated algorithm obtains a weight precision and a weight recall equal to 0.996;
- the BayesNet algorithm obtains a weight precision and a weight recall equal to 0.922 and to 0.913;
- the jRip algorithm obtains a weight precision and a weight recall equal to 0.994;
- the OneR algorithm obtains a weight precision equal to 0.874 and a weight recall equal to 0.796;
- the RepTree algorithm obtains a weight precision and a weight recall equal to 0.995.

The algorithm able to obtain the best performances is the J48 one, able to obtain a precision and a recall equal to 0.998.

**RQ2 Response:** The considered features obtain very high performances (a value precision and recall equal to 0.998) in discriminating between the several services used by the users when use the Tor network.

Relating to the time performance analyzed, all the algorithms involved in the evaluation employed a time ranging between 3.85 seconds (with the J48 algorithm) and 4.89 with the REPTree one) in the network kind identification.

IV. CONCLUSION AND FUTURE WORK

In this paper we propose a machine learning technique able to identify whether an user is using the Tor network. As secondary result, we evaluate the effectiveness of the proposed technique in the discrimination of the kind of service used in the Tor network. We apply state-of-the-art machine learning algorithms in order to evaluated our method on real-world data. We demonstrate that the considered network-based features are able to identify the Tor traffic with a precision equal to 1 and a recall equal to 1 when the classification is performed using the jRip algorithm while, a precision and a recall equal to 0.998 is obtained in the service classification. As future work we plan to follow three distinct directions: (i) investigating whether formal methods techniques can be useful in order to achieve better performances; (ii) extending the target application environment to Clouds (e.g., [28]); (iii) studying how synopsis-oriented paradigms, like those proposed in different-but-related contexts (e.g., [29], [30]), may contribute to make our traffic analysis framework more efficient.

ACKNOWLEDGMENT

This work has been partially supported by H2020 EU-funded projects NeCS and C3ISP and EIT-Digital Project Hi.

REFERENCES


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCC</th>
<th>ROC Area</th>
<th>PRC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>0.996</td>
<td>0.000</td>
<td>0.995</td>
<td>0.996</td>
<td>0.995</td>
<td>0.998</td>
<td>0.998</td>
<td>0.994</td>
<td>AUDIO</td>
</tr>
<tr>
<td>J48Consolidated</td>
<td>0.993</td>
<td>0.002</td>
<td>0.971</td>
<td>0.993</td>
<td>0.982</td>
<td>0.981</td>
<td>1.000</td>
<td>0.995</td>
<td>AUDIO</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.869</td>
<td>0.020</td>
<td>0.772</td>
<td>0.869</td>
<td>0.818</td>
<td>0.804</td>
<td>0.990</td>
<td>0.867</td>
<td>AUDIO</td>
</tr>
<tr>
<td>JRip</td>
<td>0.986</td>
<td>0.001</td>
<td>0.978</td>
<td>0.980</td>
<td>0.989</td>
<td>0.989</td>
<td>1.000</td>
<td>0.997</td>
<td>FILE-TRANSFER</td>
</tr>
<tr>
<td>OneR</td>
<td>0.994</td>
<td>0.002</td>
<td>0.994</td>
<td>0.995</td>
<td>0.995</td>
<td>0.995</td>
<td>1.000</td>
<td>0.999</td>
<td>VOIP</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.986</td>
<td>0.000</td>
<td>0.996</td>
<td>0.996</td>
<td>0.996</td>
<td>0.996</td>
<td>0.996</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>J48</td>
<td>0.998</td>
<td>0.000</td>
<td>0.999</td>
<td>0.998</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.998</td>
<td>BROWSING</td>
</tr>
<tr>
<td>J48Consolidated</td>
<td>0.993</td>
<td>0.003</td>
<td>0.988</td>
<td>0.991</td>
<td>0.990</td>
<td>0.988</td>
<td>0.997</td>
<td>0.993</td>
<td>BROWSING</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.850</td>
<td>0.024</td>
<td>0.897</td>
<td>0.850</td>
<td>0.873</td>
<td>0.846</td>
<td>0.986</td>
<td>0.948</td>
<td>BROWSING</td>
</tr>
<tr>
<td>JRip</td>
<td>0.972</td>
<td>0.012</td>
<td>0.905</td>
<td>0.912</td>
<td>0.908</td>
<td>0.896</td>
<td>0.987</td>
<td>0.957</td>
<td>CHAT</td>
</tr>
<tr>
<td>OneR</td>
<td>0.921</td>
<td>0.010</td>
<td>0.747</td>
<td>0.821</td>
<td>0.782</td>
<td>0.775</td>
<td>0.985</td>
<td>0.878</td>
<td>CHAT</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.999</td>
<td>0.000</td>
<td>0.998</td>
<td>0.998</td>
<td>0.999</td>
<td>0.999</td>
<td>1.000</td>
<td>0.999</td>
<td>MAIL</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.893</td>
<td>0.000</td>
<td>0.998</td>
<td>0.983</td>
<td>0.990</td>
<td>0.989</td>
<td>0.991</td>
<td>0.989</td>
<td>P2P</td>
</tr>
<tr>
<td>JRip</td>
<td>0.803</td>
<td>0.008</td>
<td>0.925</td>
<td>0.800</td>
<td>0.858</td>
<td>0.845</td>
<td>0.987</td>
<td>0.935</td>
<td>FILE-TRANSFER</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.973</td>
<td>0.001</td>
<td>0.998</td>
<td>0.973</td>
<td>0.986</td>
<td>0.979</td>
<td>0.996</td>
<td>0.995</td>
<td>VOIP</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.912</td>
<td>0.000</td>
<td>0.922</td>
<td>0.913</td>
<td>0.915</td>
<td>0.905</td>
<td>0.991</td>
<td>0.958</td>
<td></td>
</tr>
<tr>
<td>OneR</td>
<td>0.997</td>
<td>0.003</td>
<td>0.994</td>
<td>0.997</td>
<td>0.995</td>
<td>0.995</td>
<td>0.999</td>
<td>0.995</td>
<td>FILE-TRANSFER</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.994</td>
<td>0.002</td>
<td>0.994</td>
<td>0.994</td>
<td>0.993</td>
<td>0.993</td>
<td>0.998</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.991</td>
<td>0.001</td>
<td>0.982</td>
<td>0.991</td>
<td>0.986</td>
<td>0.985</td>
<td>0.995</td>
<td>0.974</td>
<td>AUDIO</td>
</tr>
<tr>
<td>JRip</td>
<td>0.986</td>
<td>0.001</td>
<td>0.998</td>
<td>0.986</td>
<td>0.992</td>
<td>0.990</td>
<td>0.993</td>
<td>0.986</td>
<td>BROWSING</td>
</tr>
<tr>
<td>OneR</td>
<td>0.794</td>
<td>0.000</td>
<td>1.000</td>
<td>0.794</td>
<td>0.885</td>
<td>0.888</td>
<td>0.897</td>
<td>0.801</td>
<td>CHAT</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.999</td>
<td>0.000</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>FILE-TRANSFER</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.999</td>
<td>0.000</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>FILE-TRANSFER</td>
</tr>
<tr>
<td>JRip</td>
<td>0.999</td>
<td>0.000</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>FILE-TRANSFER</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.994</td>
<td>0.002</td>
<td>0.994</td>
<td>0.994</td>
<td>0.993</td>
<td>0.993</td>
<td>0.998</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.996</td>
<td>0.000</td>
<td>1.000</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>0.999</td>
<td>0.998</td>
<td>BROWSING</td>
</tr>
<tr>
<td>JRip</td>
<td>0.994</td>
<td>0.001</td>
<td>0.997</td>
<td>0.994</td>
<td>0.996</td>
<td>0.995</td>
<td>0.999</td>
<td>0.995</td>
<td>CHAT</td>
</tr>
<tr>
<td>OneR</td>
<td>0.999</td>
<td>0.000</td>
<td>1.000</td>
<td>0.999</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>MAIL</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.994</td>
<td>0.002</td>
<td>0.994</td>
<td>0.993</td>
<td>0.992</td>
<td>0.992</td>
<td>1.000</td>
<td>0.993</td>
<td>P2P</td>
</tr>
<tr>
<td>BayesNet</td>
<td>0.994</td>
<td>0.000</td>
<td>0.997</td>
<td>0.994</td>
<td>0.996</td>
<td>0.995</td>
<td>1.000</td>
<td>0.999</td>
<td>VOIP</td>
</tr>
<tr>
<td>JRip</td>
<td>0.995</td>
<td>0.001</td>
<td>0.995</td>
<td>0.995</td>
<td>0.995</td>
<td>0.995</td>
<td>0.999</td>
<td>0.997</td>
<td></td>
</tr>
</tbody>
</table>