A Fast Unsupervised Preprocessing Method for Network Monitoring

Martin Andreoni Lopez*†, Diogo M. F. Mattos‡, Otto Carlos M. B. Duarte*, and Guy Pujolle†

*Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ - Rio de Janeiro, Brazil
† Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, F-75005 Paris, France
‡ Universidade Federal Fluminense - (UFF) - Niteroi, Brazil

Emails: {martin,otto}@gta.ufrj.br, menezes@midiacom.uff.br, guy.pujolle@lip6.fr

***Abstract*—Identifying a network misuse takes days or even weeks, and network administrators usually neglect zero-day threats until a large number of malicious users exploit them. Besides, security applications, such as anomaly detection and attack mitigation systems, must apply real-time monitoring to reduce the impacts of security incidents. Thus, information processing time should be as small as possible to enable an effective defense against attacks. In this paper, we present a fast preprocessing method for network traffic classification based on feature correlation and feature normalization. Our proposed method couples a normalization and a feature selection algorithms. We evaluate the proposed algorithms against three different datasets for eight different machine learning classification algorithms. Our proposed normalization algorithm reduces the classification error rate when compared with traditional methods. Our Feature Selection algorithm chooses an optimized subset of features improving accuracy by more than 11% within a 100-fold reduction in processing time when compared to traditional feature selection and feature reduction algorithms. The preprocessing method is performed in batch and streaming data, being able to detect concept-drift.**

I. Introduction

Maintaining the stability, reliability, and security of computer networks implies understanding type, volume and intrinsic features of flows comprising carried traffic. Efficient network monitoring allows the administrator to achieve a better understanding of the network [1]. Nevertheless, the process of network monitoring varies in complexity levels, from a simple collection of link utilization statistics, up to complex upper-layer protocol analysis to achieve network intrusion detection, network performance tuning and protocol debugging. Current network analyzing tools, such as `tcpdump`, `NetFlow`, `SNMP`, `Bro IDS` [2], Snort [3], among others, are mainly executed in one server, and the systems are not able to cope with ever-growing traffic throughput [4]. Bro IDS, for example, evolved to be used in a cluster to reach up to 10 Gb/s. Another example is the Suricata IDS, which was proposed as an improvement of the Snort IDS, using Graphical Processing Units (GPUs) to parallelize packet inspection.

In traffic analyzing and network monitoring applications, data arrives in real time from different heterogeneous sources [5], such as packets captured over the network, or multiple kinds of logging from systems and applications. An infinite sequence of events characterizes real-time stream applications, representing a continuously arriving sequence of tuples [6]. This kind of applications generates a significant volume of data. Even moderate speed networks generate huge amounts of data, for instance, monitoring a single gigabit Ethernet link, running at 50% of utilization, generates a terabyte of data in a couple of hours.

One way to attain data processing optimization is to employ Machine Learning (ML) methods. ML methods are well suited for big data since with more samples to train methods tend to have higher effectiveness [7]. Nevertheless, with high dimensional data machine learning methods tend to perform with high latency, due to computational resources consumption. For example, the K-Nearest Neighbours (K-NN) algorithm uses Euclidean distance to calculate the similarity between points. Nonetheless, Euclidean distance is unrepresentative in high dimensions. Thus, it is necessary to apply other similarity metrics, such as cosine distance, which have a higher computational cost, introducing more latency to the preprocessing. The high latency is a drawback for machine learning methods, as they must analyze data as fast as possible to reach near real-time responses. Feature Selection is a process to overcome the drawback, reducing the number of dimensions or features to a smaller subset than the set of original ones [8]. Whereas traditional Machine learning methods, also known as batch processing, deal with non-continuous data, in-stream processing Machine Learning methods, data continuously arrive, one sample at a time. Stream processing methods must process data under strict constraints of time and resources [9]. When data continuously arrive, changes in the distribution of the data as well as the relationship between input and output variables are observed over the time. This behavior, known as concept-drift [10], deteriorates the accuracy and the performance of machine learning models. As a consequence, in case of occurrence of concept-drift, we must train a new model must.

Streaming data arrives continuously form different heterogeneous source and frequently contain duplicated or missing information producing incorrect or misleading statistics [11]. Data preprocessing deals with detecting and removing errors and inconsistencies from data to improve the quality of data. The purpose of data preprocessing is to prepare the data set that will be used to fit the model. In general, the preprocessing step consumes 60% to 80% of the time of the entire machine learning process and provides essential information that will guide the analysis and adjustment of the models [12].

The number of input variables is often reduced before apply machine learning methods to minimize the use of computational resource and to improve Machine learning metrics. The variable reduction can be made in two different ways, by feature selection or by dimensionality reduction. On the one hand, feature selection only keeps the most relevant variables from the original dataset, creating a new subset of features from the originals. On the other hand, dimensionality reduction exploits the redundancy of the input data by finding a smaller set of new synthetic variables, each being a linear, or non-linear combination of the input variables.

In this paper, we present a preprocessing data method for network traffic monitoring. First, we propose a fast and efficient feature-selection algorithm. Our algorithm performs feature selection in an unsupervised manner, i.e., with no *a priori* knowledge of the output classes of each flow. We compare the presented algorithm with three traditional methods: ReliefF [13], Sequential Feature Selection (SFS), and the Principal Component Analysis (PCA) [14]. Our algorithm shows better accuracy fulfillment with the used Machine Learning methods, as well as a reduction in the total processing time. Second, we propose a normalization algorithm for streaming data, which can detect and to adapt to concept-drift.

The remainder of the paper is organized as follows. In Section II, we introduce the concept of data preprocessing. We present our preprocessing method in Section III. In Section IV, we validate our proposal and show the results. Section V presents the related work. Finally, Section VI concludes the work.

II. Data Preprocessing

Data preprocessing is the most time-consuming task in machine learning [15]. As shown in Figure 1 Data preprocessing is composed of four main steps [16]. The first step, Data Consolidation, several sources generate data; the system collects the data, and specialists interpret the data for better understanding. The second step, Data Cleaning, the system analyzes all samples and verifies if there are values that are empty or missing and is an anomaly in the dataset, also this step check if there are some inconsistencies. In the third step, Data Transformation, different functions are applied to data to improve the machine learning process. Data Normalization, this step converts variables from categorical into numerical values. In the last step, Data Reduction, techniques such as feature selection are applied to reduce data to improve and fast machine learning process. As the entire process finishes, data is ready for input in any Machine Learning algorithm. In this work, we focus on the last two steps Data Transformation and Data Reduction which are the most time-consuming steps.

Furthermore, all Feature Selection algorithms assume that data arrive preprocessed. Normalization, also known as feature scaling, is an essential method for proper use of classification algorithms because normalization bounds the domain of each feature to a known interval. If the dataset features have different scales, they may impact in different ways on the performance of the classification algorithm. Ensuring normalized feature values, usually in $[-1, 1]$; implicitly weights all features equally in their representation. Classifier algorithms that calculate the distance between two points, e.g., K-NN and K-Means, suffer from the weighted feature effect [17]. If one of the features has a broader range of values, this feature will profoundly influence the distance calculation. Therefore, the range of all features should be normalized, and each feature contributes approximately proportionately to the final distance.

Besides, many preprocessing algorithms consider that data is statically available before the beginning of the learning process [18].

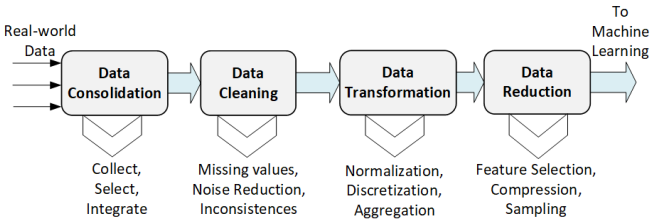

Figure 1. Preprocessing steps composed of Data Consolidation, Data Cleaning, Data Transformation and Data Reduction. Data Transformation and Data Reduction are the most time-consuming steps.

Although dealing with high dimensional data is computationally possible, the more the number of dimensions increases, the more the computational costs are challenging, and the more machine learning techniques are ineffective. Most machine learning techniques are ineffective in handling high dimensional data because they incur in overfitting while classifying data. As a consequence, if the number of input variables[1] is reduced before running a machine learning algorithm, the algorithm can achieve better accuracy. We achieve the desired variable number reduction in two different ways, dimensionality reduction, and feature selection. Dimensionality reduction exploits the redundancy of the input data by computing a smaller set of new synthetic features, each being a linear or non-linear combination of the data features. Hence, in this case, the set of input variables is a subset of synthetic features that better describe the data than the original set of features. On the other hand, feature selection only keeps the most relevant variables from the original dataset, creating a new subset of features from the original one. Thus, the set of variables is a subset of the set of features.

Dimensionality reduction is the process of deriving a set of degrees of freedom to reproduce most of a data set variability [19]. Ideally, the reduced representation should have a dimensionality that corresponds to the data intrinsic dimensionality. The data intrinsic dimensionality is the minimum number of parameters to account for the observed properties of the data. Mathematically, in dimensionality reduction, given the p-dimensional random variable x = $(x_1, x_2, \ldots, x_p)$,

[1]Features refer to the original set of attributes that describe the data. Variables refer to the input of the machine learning algorithms applied over the data. If no preprocessing method handles the original data, the set of variables and the set of features are the same.

we calculate a lower dimensional representation of it, s = $(s_1, s_2, \ldots, s_k)$ with $k \leq p$.

Algorithms with different approaches have been developed to reduce dimensionality, classified into two linear, or non-linear. The linear reduction of dimensionality is a linear projection, in which the p-dimensional data are reduced in k-dimensional data using k linear combinations of p original features. Two important examples of linear dimension reduction algorithms are principal component analysis (PCA) and independent component analysis (ICA). The goal of the PCA is to find an orthogonal linear transformation that maximizes the feature variance. The first PCA base vector, the main direction, best describes the variability of the data. The second vector is the second-best description and must be orthogonal to the first and so on in order of importance. On the other hand, the goal of ICA is to find a linear transformation, in which the base vectors are statistically independent and not Gaussian, i.e., the mutual information between two features in the new vector space is equal to zero. Unlike PCA, the base vectors in ICA are neither orthogonal nor ranked in order. All vectors are equally important. PCA is usually applied to reduce the representation of the data. On the other hand, the ICA is usually used to obtain feature extraction, identifying and selecting the features that best suit the application.

Dimensionality reduction techniques lack expressiveness since the generated features are combinations of other original features. Hence, the meaning of the new synthetic feature is lost. When there is a need for an interpretation of the model, for example, when creating rules in a firewall, it is necessary to use other methods. The **feature selection** produces a subset of the original features, which are the best representatives of the data. Thus, there is no loss of meaning. There are three types of feature selection techniques [8], wrapper, filtering and embedded.

Wrapper methods, also called closed loop, uses different classifiers, such as Support Vector Machine (SVM), Decision tree, among others, to measure the quality of a feature subset without incorporating knowledge about the specific structure of the classification function. Thus, the method will evaluate subsets based on the accuracy of the classifier. These methods consider the feature selection as a search problem, creating a NP-hard problem. An exhaustive search in the full dataset must be done to evaluate the relevance of the feature. Wrapper methods tend to be more accurate than the filtering methods, but they present a higher computational cost [20]. One popular Wrapper Method is the Sequential Forward Selection (SFS) for its simplicity. The algorithm begins with an empty set S and the full set of all features X. The SFS algorithm does a bottom-up search and gradually adds features, selected by an evaluation function, to S, minimizing the mean square error (MSE). Each iteration, the algorithm selects the feature to be included in S from the remaining available features of X. The main disadvantage of SFS is adding a feature to the set S prevents the method to remove the feature if it has the smallest error after adding others.

Filtering methods are computationally lighter than wrapper methods and avoid overfitting. Filtering methods also called open-loop methods, use heuristics to evaluate the relevance of the feature in the dataset [21]. As its name implies, the algorithm filters feature that does fill the heuristic criterion. One of the most popular filtering algorithms is the Relief. The Relief algorithm associates each feature with a score, which is calculated as the difference between the distance from the nearest example of the same class and the nearest example of the other class. The main drawback of this method is the obligation of labeling data records in advance. Relief is limited to problems with just two classes, but ReliefF [13] is an improvement of the Relief method that deals with multiples classes using the technique of the k-nearest neighbors.

Embedded methods have a behavior similar to wrapper methods, using a classifier accuracy to evaluate the feature relevance. However, embedded methods make the feature selection during the learning process and use its properties to guide feature evaluation. Therefore, this modification reduces the computational time of wrapper methods. Support Vector Machine Recursive Feature Elimination (SVM-RFE) firstly appears in gene selection for cancer classification [22]. The algorithm ranks features according to a classification problem based on the training of a Support Vector Machine (SVM) with a linear kernel. The feature with the smallest ranking is removed, according to a criterion w, in sequential backward elimination manner. The criterion w is the value of the decision hyperplane on the SVM.

All the previous feature selection algorithms are supervised methods, in which the label of the class is presented before the preprocessing step. In applications such as network monitoring and threat detection, network streams reach classifiers without a known label. Therefore, unsupervised algorithms must be applied.

III. The Proposed Preprocessing Method

Our preprocessing method comprises two algorithms. First, a normalization algorithm enforces data to a normal distribution re-scaling the values in a range between -1 and 1 interval. Indeed, the largest value for each attribute is 1 and the smallest value is -1. Normalization and standardization methods such as Max-Min or Z-score output values in a known range, usually $[-1, 1]$ or $[0, 1]$. Our normalization method is parametric-less. Then, we propose a feature selection algorithm based on the correlation between pairwise features. The Correlation Features Selection (CFS) [23] inspires the proposed algorithm. CFS scores the features through the correlation between the feature and the target class. The CFS algorithm calculates the correlation between pairwise features and the target class to get the importance of each feature. Thus, the CFS depends on target class information *a priori*, so it is a supervised algorithm. The proposed algorithm performs an unsupervised feature selection. The correlation and variance between the features measure the amount of information that each feature represents compared to others. Thus, the presented algorithm demands less computational time independently of class labeling *a priori*.

A. The proposed Normalization Algorithm

Normalization should be applied when the data distribution is unknown. Determine the distribution of streaming data is computationally expensive [24]. In our normalization algorithm 1, a histogram of a feature f_i is represented as a vector $b_1, b_2, \ldots, b_n$, such that b_k represents the number of samples that falls in the bin k.

In practice, it is not possible to know in advance the min and max for any feature. As a consequence, we use a sliding window approach, where the dataset X are the s last seen samples. For every sliding window, we obtain the min and max values for each feature. Then, data values join in a set b of intervals called bins. The idea is to divide the feature f_i in a histogram composed by bins $b_1, b_2, \ldots, b_m$, where $m = \sqrt{n} + 1$, being n the number of features, as shown in line 3 in Algorithm 1.

Each bin consists of thresholds k, for example the feature f_i is grouped in $b_1 = [min_i, k_1)$, $b_2 = [k_1, k_2)$, $\ldots$, $b_m = [k_m - 1, max_i]$. The step between threshold k is called *pivot* and it is determined as $(max_i - min_i)/m$, as it is shown in Algorithm 2. If the min or max values of the previous sliding window are smaller or bigger than min or max of the current window, that is, $min_{i-1} < min_i$ or $max_{i-1} > max_i$, new bins are created until the new values of min or max are reached. With the creation of new bins, the proposal is able to detect changes in the concept-drift.

Algorithm 1: Stream Normalization Algorithm

Input : X: Sliding window of Features, w: Window Number
Output: **H**: Normalized Features, f_r: relative frequency

```
1  if w == 1 then
2      for feature f in X do
3          b_n=√n + 1;/* n: number of features */
4          H=CreateHistogram(X,b_n);
5      end
6      else if w > 1 then
7          for sample s in f do
8              [H, f_r]=UpdateHistogram(X,b);
9          end
```

The rate between the number of observed samples in a bin and the total number of samples in the entire histogram produces the frequency of each bin. Comparing the sample x_i against the thresholds k of the bins, line 4 Algorithm 3, we define in which bin we have to increment the number of observed samples. If the value of the sample x_i is in-between the thresholds of the bin_j, then the hit number of observed samples fq_j of the bin_j is increased by one. Moreover, we calculate the relative frequency of each bin as the relation between the bin hit number and the total number of samples, $fr_j = fq_j/N$. Finally, the relative frequency values fr are mapped into a normal distribution by:

Algorithm 2: `CreateHistogram()` Function

Input : X: Sliding window of Features, b_n: number of bins
Output: **H**: Histogram

```
1  [max, min]=CalculateMaxMin;
2  k=(max − min)/(b_n); /* k: threshold */
3  for bin b in b_n do
4      b=[min_i, k);
5      k+=min;
6  end
```

Algorithm 3: `UpdateHistogram()` Function

Input : X: Sliding window of Features, b_n: number of bins
Output: **H**: Histogram, f_r: relative frequency

```
1  for sample s in X do
2      for b in bin do
3          if s in b then
4              b+ = 1;/* getting frequency */
5          else if then
6              add bin to b until s in b
7      end
8      f_r=Calculate using Equation 1;
9      H=map s to NormalDistribution;
10 end
```

$$Z > P\left(x = \sum_{0}^{m} fr_j\right). \tag{1}$$

with Equation 1 is it possible to see that all values are now mapped into a normal probability distribution with $\mu = 0$ and $\sigma = 1$, line 8 in algorithm 3. As a consequence, all samples are normalized between $-1 \leq x_i \leq 1$.

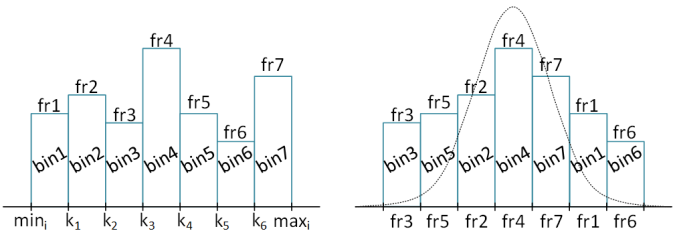

Figure 2. Representation of the feature divided in histogram. Each feature is divided in bins that represent the relative frequency of the samples comprised between the thresholds k. The second step of the algorithm approximate the histogram to a normal distribution.

If we consider the process that generates the stream is non-stationary, it implies a possible concept-drift. Haim and Tov affirm that histogram must be dynamic when dealing with streaming data [25]. As a consequence, intervals do not have a fixed value, and the bins adapt to concept-drift. However, if the bins remain static, it reflects the evolution of the change during

time [26]. In our application, feature normalization for network monitoring, we follow the former approach. Maintaining fixed intervals allow us to see how a feature evolves. Also, as our histogram algorithm creates new bins when a value does not enter in any of the current intervals, it enables to detect outliers dynamically. In streaming data, it is not possible to maintain all the samples x_i, because it is computationally inefficient and, in case of unlimited data, it does not fit in memory. Our algorithm only efficiently keeps the frequency of each bin.

The most complicated function in the normalization process is to update the bins. If the max and min reference values of the window change, the bins update functions takes the complexity $\mathcal{O}(n)$ on time. The creation of the histogram is only done in the first window and takes constant time. The histogram update uses a binary search to fill the bin value in $\mathcal{O}(\log n)$ time.

B. The proposed Correlation-Based Feature Selection

We propose the Correlation-Based Feature Selection, a simple unsupervised filter method for feature selection. Our method uses the correlation between features. The Pearson correlation of two variables is a measure of their linear dependence. The key idea of the method is to weight each feature based on the correlation of the feature against all other features that describe the dataset. We adopt the Pearson's coefficient as the correlation metric. Pearson's coefficient value is between $-1 \leq \rho \leq 1$, where 1 means that the two variables are directly correlated, linear relationship, and -1 in the case of an inverse linear relationship, also called anticorrelation.

The Pearson Coefficient ρ, can be calculated in terms of mean μ and standard deviation σ,

$$\rho(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{A_i-\mu_A}{\sigma_A}\right)\left(\frac{B_i-\mu_B}{\sigma_B}\right), \quad (2)$$

or in terms the of covariance

$$\rho(A,B) = \frac{cov(A,B)}{\sigma_A\sigma_B}, \quad (3)$$

then we calculate the weight vector,

$$\mathbf{w_i} = \frac{\sigma_i^2}{\sum_{j=0}^{j=N}|\rho_{ij}|}. \quad (4)$$

Firstly, we need to obtain the correlation matrix, calculated by Equation 3, line 1 algorithm 4. The correlation matrix is the pairwise covariance calculations between features. Then, applying the Equation 4, we establish a weight w that is a measure of the importance of the feature. To calculate w, we sum the absolute values of the correlation features, lines 5-6 algorithm 4. This absolute value sum is due to Pearson's coefficient, ρ may assume negative values. Then we calculate the variance $\hat{V}$ of each feature that privileges the feature that has greater variance and lower covariance, line 8 algorithm 4. The idea is to establish which feature represent the most information, giving the correlation between two features. Furthermore, the weights give us an indication of the amount of information the feature has independently from the others. The weight w has values between $0 \leq N$, where N is the number of features and 0 means that the features are independent of the other. The higher the w value is, the higher is the variance of the feature and lesser correlation with other features. Thus, more information is the aggregated by this feature.

Algorithm 4: Correlation Based Feature Selection

Input : X: Matrix of Features and Data
Output: **r**: Vector of Ranked Features, **w**: Vector of weights

1 $\rho = Corr(X)$ /* Correlation Matrix */
2 **for** $0 \leq i < len\,(\rho)$ **do**
3 $W_i = 0$
4 **for** $0 \leq j < len\,(\rho_i)$ **do**
5 $k_i = abs(\rho_{ij})$ /* Absolute Values */
6 $aux_i += k_i$ /* Sample Addition */
7 **end**
8 $\mathbf{w_i} = \hat{V}(i)/aux_i$ /* Calculate Weights */
9 **end**
10 $\mathbf{r} = sort(\mathbf{w}, byhighervalues)$

IV. Evaluation

To evaluate the proposed algorithm, we perform traffic classification to detect threats. We chose the traffic classification application because it is time sensitive and our algorithm can significantly reduce the processing time, enabling prompt defense mechanisms. We implemented traffic classification using machine learning algorithms against three different datasets. The NSL-KDD is a modification of the original KDD-99 dataset and has the same 41 features and the same five classes, Denial of Service (DoS), Probe, Root2Local (R2L), User2Root (U2R) and normal, as the KDD 99 [27]. The improvements of the NSL-KDD over KDD 99 are the elimination of redundant and duplicate samples, to avoid a biased classification and overfitting, and a better cross-class balancing to avoid random selection. GTA/UFRJ dataset[2] [28] combines real network traffic captured from a laboratory and network threats produced in a controlled environment. Network traffic is abstracted in 26 features and contains three classes, DoS, probe and normal traffic. The NetOp[2] is a real data set from a Brazilian operator [29]. The dataset has anonymous access traffic of 373 broadband users of the South Zone of the city of Rio de Janeiro. We collected data during 1 week of uninterrupted data collection, from February 24 to March 4, 2017. We summarized packets in a dataset of 46 flow features, associated with an IDS alarm class or the legitimate traffic class. All datasets are summarized in Table I.

We count on Intel Xeon processors with a clock frequency of 2.6 GHz and 256 GB of RAM for conducting the measurements.

[2]Anonymized data can be asked by sending an email contact to the authors

Table I
SUMMARY OF DATASETS USED FOR EVALUATION (FE: FEATURES; FL: FLOWS).

Dataset	Format	Size	Attacks	Type
NSL-KDD	41 Fe	150k Fl	80.5 %	Synthetic
GTA/UFRJ	26 Fe	95 GB	30 %	Synthetic
NetOp	46 Fe	5M Fl	-	Real

In the first experiment, we use one day from NetOp dataset to evaluate our normalization method. Shapiro–Wilk test was used to verify that our proposal enforces a normal distribution for the normalized features. Table II shows Shapiro-Wilk test, we considered α = 0.05. We evaluate the hypothesis that our proposal normalization method follows a normal distribution. According to the results, the proposed method has a p-value of 0.24 $>$ 0.05, and W is closer to one, W = 0.93, then we assume that samples are not significantly different from a normal population. In the case of Max-Min normalization [30], p-value is very smaller than the α, and W indicates that it is far from 1. As a consequence, we refuse the hypothesis assuming than sampling data are significantly different from a normal population. Figure 3 shows a graphical interpretation of the Shapiro–Wilk test, it represents a sample after being normalized. As our proposal follows the normal distribution, the blue points follow the dashed line, while the max-min approach follows a right-skewed distribution.

Table II
HYPOTHESIS COMPARISON FOR A NORMAL DISTRIBUTION APPROACH. IN SHAPIRO-WILK TEST p-VALUE IS $0.24 > 0.05$, AND W IS CLOSER TO ONE, W=0.93, CONFIRMING THAT VALUES FOLLOW A NORMAL DISTRIBUTION.

	Shapiro-Wilk	
	Mean W	Mean p
proposal	0.93	0.24
max-min	0.65	9.28e-07

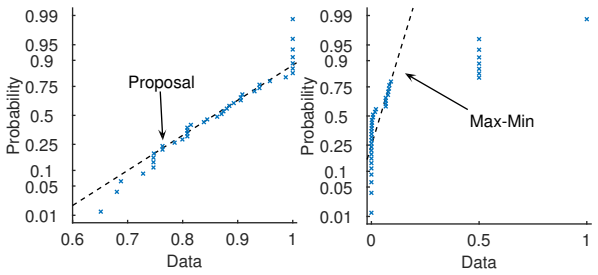

Figure 3. Representation of the feature divided in histogram. Each feature is divided in bins that represent the relative frequency of the samples comprised between the thresholds k.

In the following experiments, we verify our preprocessing method in a use case of traffic classification. Thus, we implement the Decision Tree (DT), with the C4.5 algorithm, Artificial Neural Networks (ANN), and Support Vector Machine (SVM) as classification algorithms to evaluate the proposed feature selection algorithm. We selected these algorithms because they are the most used ones for network security [31]. In all methods, we perform the training in a 70% partition of the dataset and the test run over the remaining 30%. During the training phase, we perform tenfold cross-validation to avoid overfitting. In cross-validation, parts of the dataset are divided and not used in model parameters estimation. They are further used to check whether the model is general enough to adapt to new data, avoiding overfitting to training data.

The Decision Tree Algorithm: In a decision tree, leaves represent the final class and branches represent conditions based on the value of one of the input variables. During the training part, the C4.5 algorithm determines a tree-like classification structure. The real-time implementation of the decision tree consists of if-then-else rules that generate the tree-like structure previously calculated. The results are presented in the Section IV-A, along with the ones from the other algorithms.

The Artificial Neural Network Algorithm: The artificial neural networks are inspired on the human brain, in which each neuron performs a small part of the processing and transfers the result to the next neuron. In artificial neural networks, the output represents a degree of membership for each class, and the highest degree is selected. The weight vectors Θ are calculated during the training. These vectors determine the weight of each neuron connection. In training, there are input and output sample spaces and the errors, caused by each parameter. The back-propagation algorithm minimizes the errors.

In order to determine to which class a sample belongs each neural network layer computes the following equations:

$$z_{(i+1)} = \Theta_{(i)} a_{(i)} \quad (5) \qquad a_{(i+1)} = g(z_{(i+1)}) \quad (6) \qquad g(z) = \frac{1}{1+e^{-z}} \quad (7)$$

where a is the vector that determines the output of layer i, $\Theta_{(i)}$ is the weight vector that leads layer i to layer $i+1$, and $a_{(i+1)}$ is the output of layer $i+1$. The function $g(z)$ is the activation function, represented by *Sigmoid* function, which plays an important role in the classification. For high values of z, $g(z)$ returns one and for low values $g(z)$ returns zero. Therefore, the output layer gives the degree of membership in each class, between zero and one, classifying the sample as the highest one. The activation function enables and disables the contribution of a certain neuron to the final result.

The Support Vector Machine Algorithm: The Support Vector Machine (SVM) is a binary classifier, based on the concept of a decision plane that defines the decision thresholds. SVM algorithm classifies through the construction of a hyperplane in a multidimensional space that split different classes. An iterative algorithm minimizes an error function, finding the best hyperplane separation. A kernel function defines this hyperplane. In this way, SVM finds the hyperplane with a maximum margin, that is, the hyperplane with the most significant distance possible between both classes.

The real-time detection is performed by the classification to each class pairs: normal and non-normal; DoS and non-DoS;

and probe and non-probe. Once SVM calculates the output, the chosen class is the one with the highest score. The classifier score of a sample x is the distance from x to the decision boundaries, that goes from $-\infty$ to $+\infty$. The classifier score is given by:

$$f(x) = \sum_{j=1}^{n} \alpha_j y_j G(x_j, x) + b, \tag{8}$$

where $(\alpha_1, \ldots, \alpha_n.b)$ are the estimated parameters of SVM, and $G(x_j, x)$ is the used kernel. In this work, the kernel is linear, that is, $G(x_j, x) = x_j^{'} x$, which presents a good performance with the minimum quantity of input parameters.

A. Classification Results

This experiment shows the efficiency of our feature selection algorithm when compared to literature methods. We try a linear Principal Component Analysis (PCA), The ReliefF algorithm, the Sequential Forward Selection (SFS), and the Support Vector Machine Recursive Feature Elimination (SVM-RFE). For all methods, we analyze their version with four and six output features in the GTA/UFRJ dataset. For the sake of fairness, we tested all the algorithms with the classification methods presented before. We use a decision tree with the with a minimum of 4096 leaves, a binary support vector machine (SVM) with linear kernel, and finally a neural network with one hidden layer with 10 neurons. We use ten-fold cross-validation for our experiments.

Figure 4 presents information gain (IG) sum of the selected feature for each evaluated algorithm. Information gain measures the amount of information, in bits, that a feature adds compared to the class prediction. Thus, information gain calculation computes the difference of target class entropy and the conditional entropy of the target class given the feature value as known. When employing six features, the results show our algorithm has information retention capability between SFS and ReliefF, and higher than SVM-RFE. The information retention capability of PCA, is higher than feature selection methods, as each feature is a linear combination of the original features and is computed to retain most of the dataset variance.

Figure 5 shows the accuracy of the three classification methods, Decision Tree, Neural Network and Support Vector Machine (SVM), when different dimensionality reduction

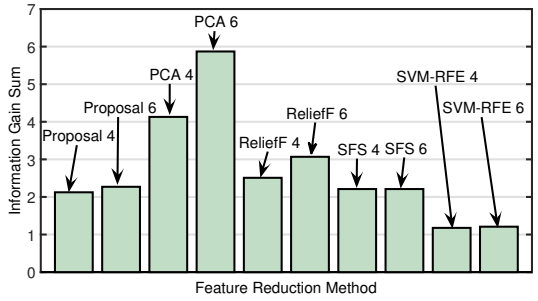

Figure 4. Information gain sum for feature selection algorithms. The selected features by our algorithm keeps an information retention capability between SFS and ReliefF in the GTA/UFRJ dataset.

methods choose the input variables. In the first group, our proposal with six features reaches 97.4% accuracy, which is the best results for the decision tree classifier. The following result is PCA with four and six features in 96% and 97.2%. The Sequential Forward Selection (SFS) presents the same result with four and six features with 95.5%. The ReliefF algorithm has the same results in both four and six features is 91.2%. Finally, the lowest result is shown by the SVM-RFE algorithm with four and six features. As the decision tree algorithm creates the decision nodes based on the variables with higher entropy, the proposed feature selection algorithm better performs because it keeps most of the variance of the dataset.

The second classifier, the neural network, the best result is shown by the PCA with six features in 97.6% of accuracy. However, the PCA with four features presents a lower performance with 85.5%. ReliefF presents the same results for both features in 90.2%. Our proposal shows a result with 83.9% and 85.0% for four and six features. On the other hand, the SFS presents the worst results of all classifiers, 78.4% with four features and 79.2% with six features. One impressive result is the SVM-RFE, with four features presents a deficient result of 73.6% that is one of the worst for all classifiers, however, with six features present almost the same best second result with 90.1%.

In the Support Vector Machine (SVM) classifier, the PCA presents a similar behavior compared with the neural networks. For six features presents the highest accuracy of all classifiers with 98.3%, but just 87.8% for four features. ReliefF again presents the same result for both cases in 91.4%. Our proposal has 84% for four features and 85% for six features. SFS present the same result for both features in 79.5%. The lowest accuracy of this classifier is the SVM-RFE with 73.6% for both cases. As our proposal maximizes the variance on the resulting features, the resulting reduced dataset spreads into the new space. For a linear classifier, such as SVM, it is hard to define a classification surface for a spread data. Thus, the resulting accuracy is not among the highest. However, as the selected set of features still being significant for defining the data, the resulting accuracy is not the worst one.

The Sensitivity metric shows the rate of correctly classified samples. It is a good a metric to evaluate the success of a classifier when using a dataset in which a class has much more samples than others. In our problem, we use sensitivity as a metric to evaluate our detection success. For this, we consider the detection problem as a binary classification, i.e., we consider two classes: normal and abnormal traffic. In this way, the Denial of Service (DoS) and Port Scanning threat classes compose a common attack class. Similar to Accuracy representation in Figure 5, Figure 6 represents the sensitivity of the classifiers applying the different methods of feature selection. In the first group, the classification with Decision Three, PCA shows the best sensitivity with 99% of correct classification, our algorithm achieves a performance of almost 95% of sensitivity, with four and six features. Neural Networks, represented in the second group, has the best

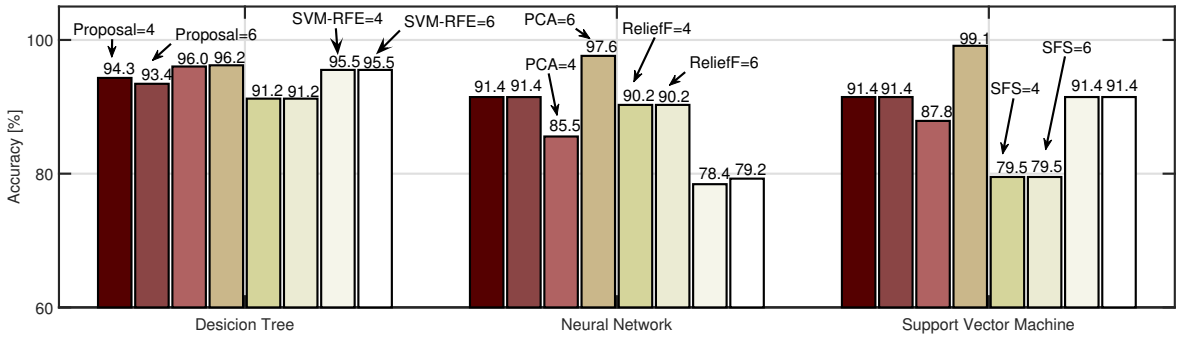

Figure 5. Accuracy comparison of features selection methods. Our Proposal, Linear PCA, ReliefF, SFS and SVM-RFE compared in decision tree, SVM, and neural network algorithms in the GTA/UFRJ dataset.

sensitivity with PCA using six features with 97.7%, then our results show good performance with both four and six features in 89%. In this group, the worst sensitivity of all classifiers is reached by the SVM-RFE with four and six features in 69.3%. Finally, the last group shows the Sensitivity for Support Vector Machine (SVM) classifier. Again, showing a similar behavior to the previous group PCA with six features shows the best sensitivity with 97.8%. Then, the second-best result is reached by our algorithm, as well as with ReliefF, with 89% sensitivity with both features. It is worthy to note that our algorithm presents a stable behavior in Accuracy as well as in Sensitivity. We highlight that our algorithm performs nearly equal to PCA. PCA creates artificial features that are a composition of all real features, while our algorithm selects some features from the complete set of features. In this way, our algorithm was the best feature-selection method among the evaluated ones, and it also introduces less computing overload when compared with PCA.

When analyzed the features each method chooses, it is possible to see none of the methods selects the set of same features. Nevertheless, ReliefF and SFS select as the second-best *amount of IP packets*. One surprising result from the SFS is the election of *Amount of ECE Flags* and *Amount of CWR Flags*. In a correlation test, these two features show that there is no information inclusion because they are empty variables. However, we realized that one of the main features is *Average Packet Size*. In this dataset, the average packet size is fundamental to classify attacks. One possible reason is that, during the creation of the dataset, an automated tool performed the Denial of Services (DoS) and Probe attacks. Mainly this automated tool produces attacks without altering the length of the packet.

Figure 7 shows a comparison of the processing time of all implemented feature selection and dimensionality reduction methods. All measures are in relative value. We can see that SFS presents the worst performance. The SFS algorithm performs multiple iterations to minimize the mean square error (MSE). Consequently, all these iterations increase the processing time. Our proposal shows the best processing time together with PCA because both implementations perform matrix multiplication. Matrix multiplication is a simple computation function.

The next experiment is to evaluate our proposal in different datasets. We use the NSL-KDD dataset and the NetOp dataset. Besides linear SVM, Neural Network and Decision Tree, we also evaluate K-Nearest Neighbors (K-NN), Random Forest, two kernels, linear and Radial Basis Function (RBF) kernel in Support Vector Machine (SVM), Gaussian Naive Bayes, and Stochastic Gradient Descendant. Adding these algorithms, we cover the full range of the most important algorithms for supervised machine learning.

The **Random Forests** (RF) algorithm avoids overfitting when compared to the simple decision tree because it constructs several decision trees, trained in different parts of the same dataset. This procedure decreases the variance of classification and improves the performance regarding the classification of a single tree. The prediction of the class in the RF classifier consists of applying the sample as input to all the trees, obtaining the classification of each one of them and, then, a voting system decides the resulting class. The construction of each tree must follow the rules: (i) for each node d, select k input variables of total m input variables, such that $k \ll m$; to calculate the best binary division of the k input variables for the node d, using an objective function; repeat the previous steps until each tree reaches l number of nodes or until its maximum extension.

The simple Bayesian classifier (**Naive Bayes** - NB) takes the strong premise of independence between the input variables to simplify the classification prediction, that is, given the value of each input variable, it does not influence the value of the others input variables. From this, the method calculates the probabilities *a priori* of each input variable, or a set of them, to set up a given class. As a new sample arrives, the algorithm calculates for each input variables the probability of being a sample of each class. The output of all probabilities of each input variable results in *a posterior* probability of this sample belonging to each class. The algorithm, then, returns the classification that contains the highest estimated probability.

In the **K-Nearest Neighbors** (K-NN) the class definition

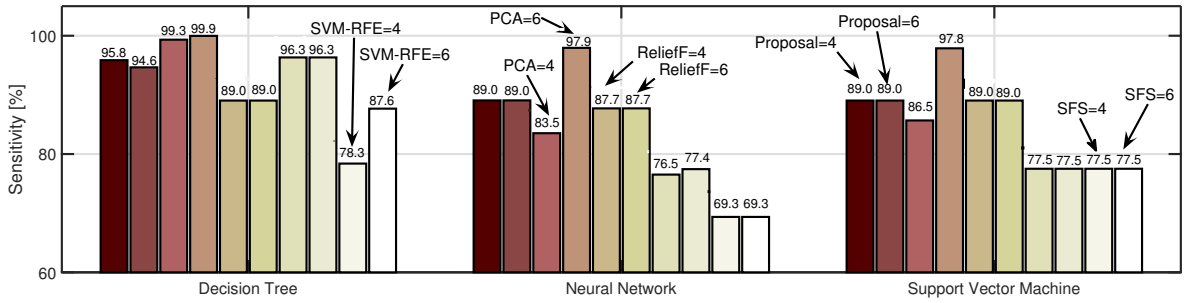

Figure 6. Sensitivity of detection in decision tree, SVM, and neural network algorithms for feature selection methods in the GTA/UFRJ dataset.

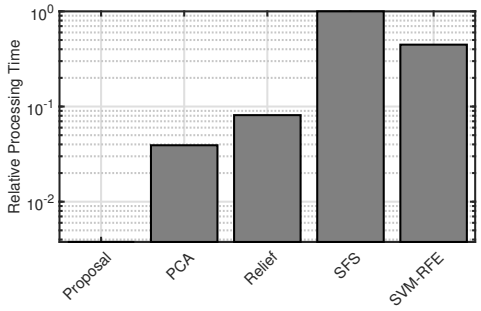

Figure 7. Performance of features selection algorithms according to processing time. Our proposal and the PCA present the best processing time in the GTA/UFRJ dataset.

of an unknown sample is based on the k-neighbors classes closest to the sample. The value k is a positive integer and usually small. If $k = 1$, then the sample class is assigned to the class of its nearest neighbor. If $k > 1$, the sample class is obtained by starting a resultant function, such as a simple voting or weighted voting, of the k-neighbors' classes. The neighborhood definition uses a measure of similarity between samples in the feature space. Threat detection literature often use Euclidean distance, although other distances have good results and the best choice for a similarity measure will depend on the type of the used dataset [32]. The Euclidean distance of two samples $\mathbf{p}$ and $\mathbf{q}$ in the space of n features is given by

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i}^{n} (p_i - q_i)^2}. \quad (9)$$

Stochastic Gradient Descent with Momentum: This scheme relies on the Stochastic Gradient Descent (SGD) [33] algorithm, is a stochastic approximation of Gradient Descent, in which a single sample approximates the gradient. In our application, we consider two classes, normal and threat. Therefore, we use the Sigmoid Function, expressed by

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\intercal x}}, \quad (10)$$

to perform logistic regression. In the Sigmoid function, low values of the parameters $\theta^\intercal$ times the sample feature vector x return zero, whereas high values return one. When a new sample $x_{(i)}$ arrives, the SGD evaluates the Sigmoid function and returns one for $h_\theta(x_{(i)})$ greater than 0.5 and zero otherwise. This decision presents an associated cost, based on the real class of the sample $y_{(i)}$. The cost function is defined in Equation 11. This function is convex, and the goal of SGD algorithm is to find its minimum, expressed by

$$J_{(i)}(\theta) = y_{(i)} log(h_\theta(x_{(i)})) + (1 - y_{(i)}) log(1 - h_\theta(x_{(i)})). \quad (11)$$

On each new sample, the algorithm takes a step toward the cost function minimum based on the gradient of the cost function.

Validation in NSL-KDD and NetOp Datasets: The first experiment evaluates the performance of the feature selection in both datasets. In this experiment, we vary the number of selected features to evaluate the impact on the accuracy. We analyze the performance with no feature selection (No FS), and then we reduce features from 10% to 90% of the original set of features. All the experiments were performed using a K-fold cross-validation. The K-fold cross-validation performs K training iterations in the partitions of the data and, at each iteration, in the remaining $K - 1$ partition, the K-fold cross-validation performs the test in a mutually exclusive manner. We use $K = 10$, which is the commonly used value. Figure 8 shows the effect of feature selection. No Feature Selection performs well for almost all algorithms. Reducing the number of features in 10%, however, improve accuracy in all algorithms, except for Random Forest. In contrast, a bigger reduction of feature deteriorates the accuracy for all classifiers.

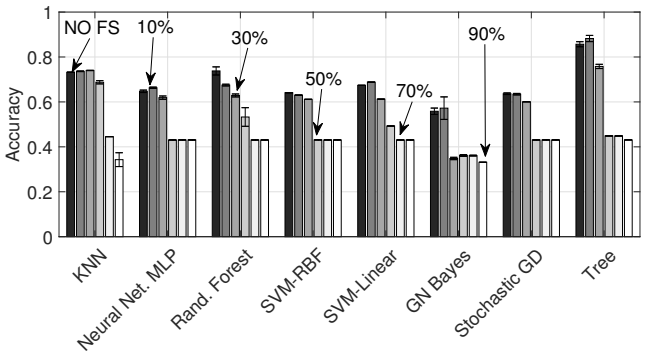

Figure 8. Evaluation of Feature Selection varying the selected features in NSL-KDD dataset.

We also measure others metrics, such as Precision, Sensitivity, F1-Score, training and classification time. We compare the effect of 10% reduction in all these metrics. Figure 910 show accuracy, precision, sensitivity and F1 - score for dataset with no feature selection Figure 9 and with 10% of reduction Figure 10. For K-NN, SVM with Radial Basis Function (RBF) kernel, and Gaussian Naive Bayes metrics remain the same. For the Neural Network, MLP and SVM with a linear kernel, an improvement between 2-3% in all metrics is reached with 10% of features reduction. Random Forest present the worst performance when features are reduced; all metrics worsen their values between 8-9%. Stochastic Gradient Descendant (SGD) also suffer a small reduction of 1% in their metrics. The decision tree is the most benefited improving between 3-4% their metrics, which shows the capability of reducing overfitting when applying feature selection.

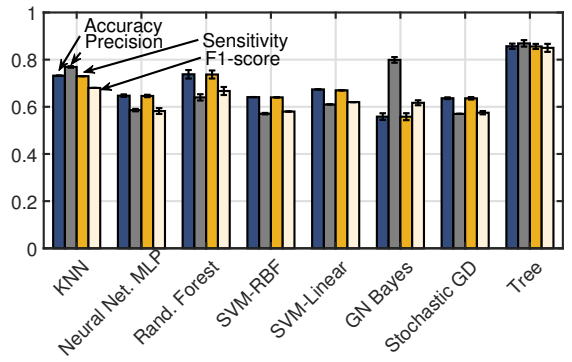

Figure 9. Accuracy, precision, sensitivity and F1-Score for NSL-KDD. Metrics with no future selection.

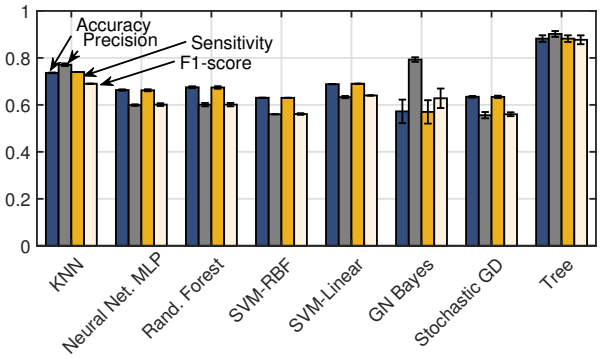

Figure 10. Metrics reducing only 10% of the initial features in the NSL-KDD dataset.

Figure 11 shows training and classification time with no features selection, while Figure 12 shows results with 10% of reduced features. The K-NN algorithm training time augmented considerably, passing from 0.63 seconds to 5.03, while classification time also suffers an increase passing from 1.89 seconds to 2.88. Neural Network reduced 9% of the training time, from 22.99 seconds to 20.92, classification time got 0.01 second increased. Random Forest training time increased 0.02

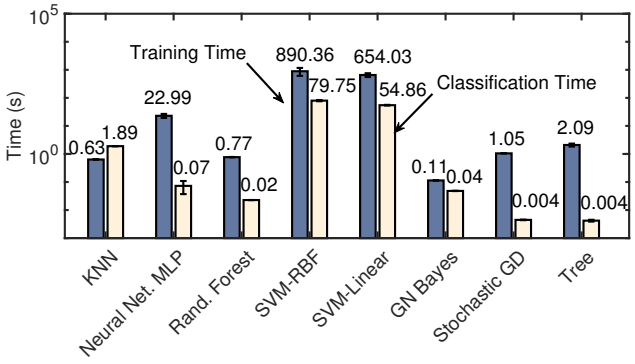

Figure 11. Classification and training time in NSL-KDD Dataset with no feature Selection

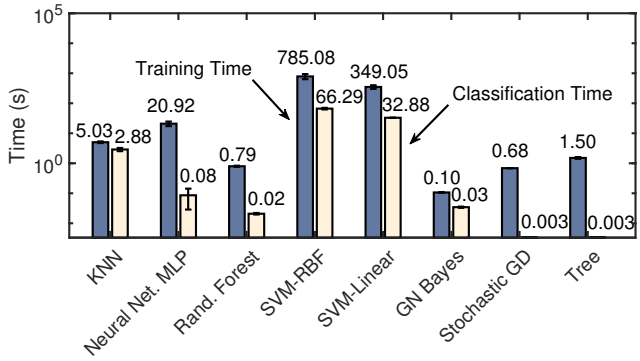

Figure 12. Classification and training time in NSL-KDD Dataset with only 10% of the initial features

second, and classification time remained the same, which is negligible because of the intrinsic error of the cross-validation. SVM with Radial Basis Function (RBF) and SVM with linear kernel are the most benefited from features selection. SVM-RBF training time reduced 11% while the classification time, 16%. SVM-Linear classification time reduced 46%, from 654 seconds to 349 seconds, and training time, 40%, from 54.86 to 32.88 seconds. Feature selection in Gaussian Naive Bayes, Stochastic Gradient Descendant, and Decision Tree strongly impact in training time with an approximate reduction of 30%, while the classification time was reduced of a one-time unit in three algorithms.

We performed the same experiment in the NetOp Dataset. Figure 13 shows the accuracy of different classifiers while reducing from 10% to 90% of the features. Using the NetOp dataset, applying feature selection keeps unaffected classifier accuracy unaffected. In the case of K-NN, the accuracy variation is less than 0.02%. A similar case occurs with Neural Networks, SVN with linear and with RBF kernels, Stochastic Gradient Descendant, and Decision Tree. In Random Forest, the best accuracy is found with a reduction of 30% of the original set of features of the dataset. The best result is reached in Gaussian Naive Bayes, in which 90% of the reduction in the selected features increases the accuracy from 57% to 78%,

using only five features.

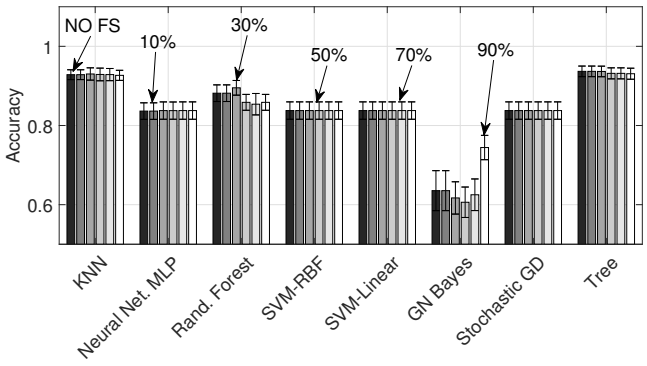

Figure 13. Evaluation of Feature Selection varying the selected features in NetOp dataset.

Reducing 90% of selected features, we analyze other metrics, such as Precision, Sensitivity, and F1-Score, for all classifiers. We compare the results with no feature selection, Figure 14, and with only five features, Figure 15. All metrics remains almost equal. We achieve a slight positive variation in Gaussian Naive Bayes and Random Forest. We conclude that, for this dataset, our Feature Selection method maintains the metrics unaltered or increase classifier performance, because our proposal keeps the most of independent features in the dataset.

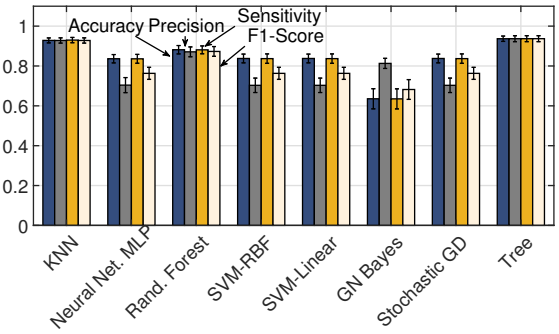

Figure 14. Accuracy, precision, sensitivity and F1-Score for NetOp dataset. Metrics with no future selection.

Figure 16 shows the training and classification times with no feature selection, while Figure 16 shows the training and classification times for the dataset with 90% of feature reduction. All the classifiers reduced their times. K-NN training time is reduced by 71%, while classification time is reduced by 84%. For Neural Networks reduced the training time by 25% and classification time is reduced in 0.02 seconds. Random Forest reduced their training time by 38% while their classification time remains the same. SVM with RBF kernel training time is reduced by 78% and training time is reduced by 54%. SVM with linear kernel received the biggest improvement. Training time was reduced by 88% while classification time was reduced by 81%. Gaussian Naive Bayes reduced their

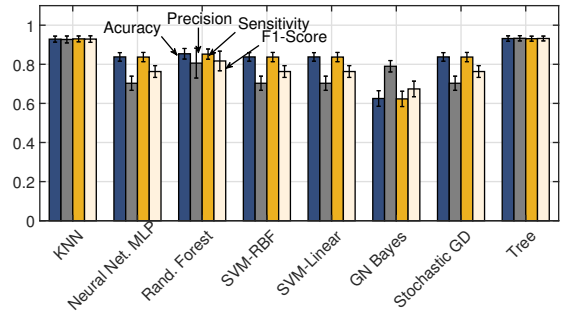

Figure 15. Metrics reducing only 90% of the initial features in the NetOp dataset.

training time in 80% while classification time was reduced in 76%. Stochastic Gradient Descendant also shows a reduction of 61% in training and 66% for classification time. Finally, Decision Tree reduced training time by 79% and classification time got faster, being reduced by 28%. As a consequence, a feature reduction of 90% impacts directly in the training and classification time of the machine learning classifiers. Therefore, our Feature Selection method improves training and classification times in all classifiers.

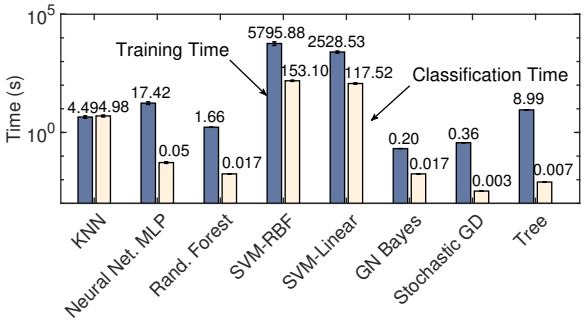

Figure 16. Classification and training time in NetOp Dataset with no feature Selection

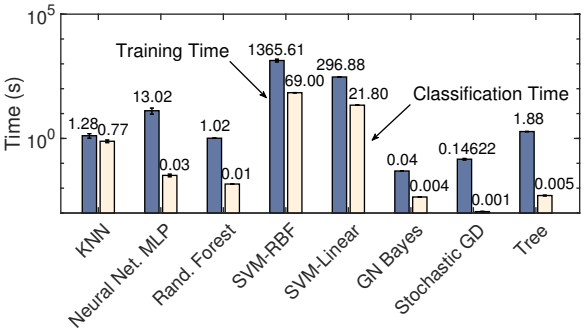

Figure 17. Classification and training time in NetOp Dataset with only 90% of the initial features

In this experiment, we show the most important group of features. Thus, we group features into eight groups in the NetOp dataset. We remove the flow tuple information features

because our algorithm works on numerical features and tuple information features are categorical. Table III describes the groups. We established the window size at 1000 samples. Figure 18 shows the accuracy for all seven algorithms for classification. In Decision Tree, all groups show similar behavior and present high accuracy. Gaussian Naive Bayes and SVM with the linear kernel for group 3, Time Between Packets, and for group 5, subflow information, present the lowest accuracy. For the rest of the groups, these classifiers also reach high accuracy. K-Nearest Neighbors (K-NN) shows a special case, besides group 2, which is the highest accuracy, all the other groups show different behaviors. In Neural Networks, groups 2 and 3, Packet Statistics and Time between Packets, show the highest accuracy, while the reminding groups maintain in 50%. Random Forest shows a similar behavior than Decision Tree, with high accuracy in all their groups. Nevertheless, the group 5, subflow information, present the lowest accuracy. Stochastic Descendant Gradient shows the highest accuracy in group 2,6 and 7. We conclude that group 2, Packet Statistics, is the most important for the accuracy calculation for all the classifiers.

Table III
FEATURES GROUPS

Group	Description	Number of Features
G1	Packet Volume	4
G2	Packet Statistics	8
G3	Time Between Packets	8
G4	Flow Time Statistics	9
G5	SubFlow Information	4
G6	TCP Flags	4
G7	Bytes in headers + ToS	3

Finally, this experiment shows how our preprocessing method when executing with machine learning classifiers in stream data, can detect concept-drift. This experiment also demonstrates that the proposed preprocess method can run under batch and stream data. We use the flow diagram of the Figure 19. We force traditional learning methods to become adaptive learners to detect the concept-drift. Adaptive learners dynamically adapt to new training data when the learned concept is contradicted. Once a concept-drift is detected, a new model is created.

We validate the proposal with the NetOp dataset. The dataset presents labeled samples in threats and normal traffic, a binary classification problem. We divide the dataset in training set and test set, in a relation of 70% for training and 30% for the test. We consider the training set as static with T consecutive sample windows. We have used the Synthetic Minority class Oversampling TEchnique (SMOTE) [34] approach to oversampling the minority class, only in the training set, initial window. When the number of samples in a class is predominant in the dataset, it is called class imbalance. Class imbalance is typical in our kind of threat detection application when attacks are rare events when compared to normal traffic. The test set is streaming data arriving at the same frequency. We group data in sliding windows of N samples.

Figure 20 shows the accuracy when we analyze one day from NetOp dataset. In the experiment, we measure the impact of the concept-drift on the final accuracy. Determining the concept-drift helps to improve the performance of the system, since the model will not be recalculated. We train different static algorithm with 30% of the dataset. We use 1000 samples as window size. The trained static algorithms are the Support Vector Machine (SMV) with linear kernel, and with Radial Basis Function (RBF) kernel, Gaussian Naive Bayes, Decision Tree, and Stochastic Gradient Descent (SGD). The decision tree has the worst accuracy when compared with the other algorithms. Decision tree shows a low accuracy in the second window. This behavior means that the created model during the training step does not adequately represent the model of the entire dataset. Stochastic Gradient Descendant shows a similar behavior of decision tree, having a concept-drift in the second window. The SVM with linear kernel presents a concept-drift in the seventh window. SVM with RBF shows a lower accuracy during all experiment and a concept-drift at the last window. Finally, due to the implementation of the Gaussian Naive Bayes, it follows the same probability distribution as our normalization method, as a consequence does not present any concept-drift.

V. RELATED WORK

State-of-art proposals focus on algorithms for online feature selection. Perkins and Theiler Grafting algorithm based on a stage-wise gradient descent approach. Gradient Feature Testing (grafting) [35] treats feature selection in the core of the learning process. The objective function is a binomial negative log-likelihood loss. The Grafting method uses an incremental and iterative gradient descent. For each step, a heuristic is used to identify which feature improves the existent model. If the feature optimizes the model, the selected feature and the model are returned. The Grafting algorithm is able to operate with streaming features. A value of the regularization parameter λ must be determined in advance to establish which feature is most likely to improve the model at each iteration. To determine the value of λ is required information about the global feature set. As a consequence, Grafting method is ineffective with streaming data with unknown feature size.

The Alpha-investing method [36] considers that new features arrive in a streaming manner generated sequentially for a predictive model. The main advantage of Alpha-investing is the possibility to run in a feature set of unknown or even infinite sizes. Every time a feature arrives, alpha-investing uses linear regression to reduce the threshold of error dynamically. As a drawback, alpha-investing only considers the addition of new features without evaluating the redundancy after the feature inclusion.

Wu *et al.* presented the OSFS (Online Streaming Feature Selection) algorithm and its faster version, the Fast-OSFS algorithm, to avoid the redundancy of added features [37]. The OSFS algorithm uses a Markov blanket of a feature to determine the relevance of the feature in relation with their neighbors. The Markov blanket of a node A, $MB(A)$, is its

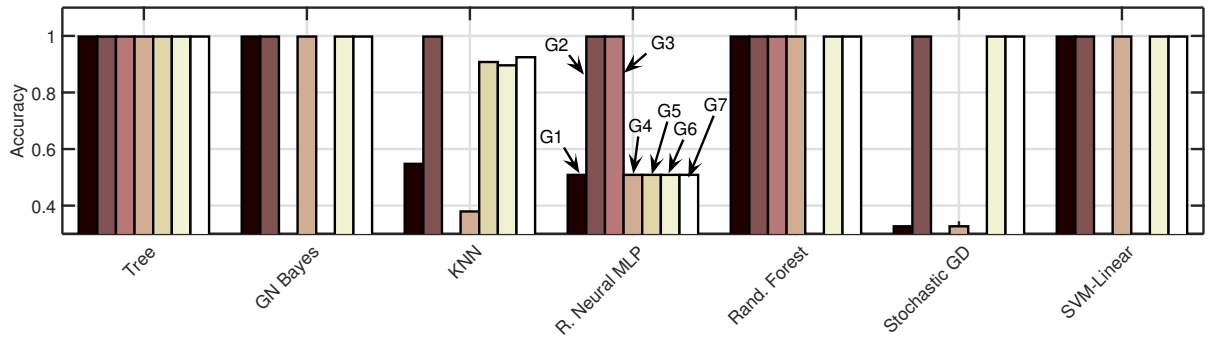

Figure 18. Evaluation of group features with different machine learning algorithms.

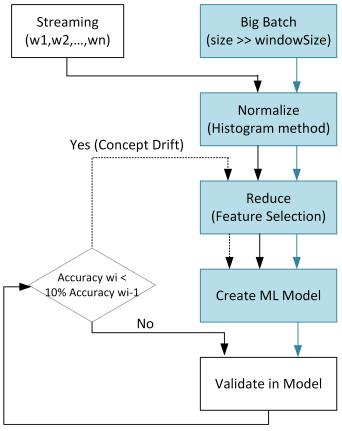

Figure 19. Flow diagram used for proposal evaluation

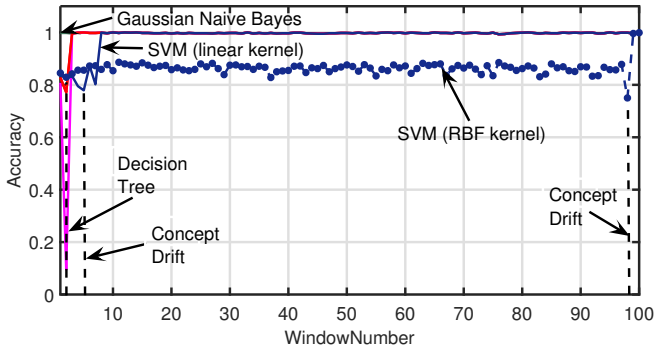

Figure 20. concept-drift detection in one day of the NetOp dataset. Our proposal was able to detect early concept-drift in SGD and in SVM with linear kernel. Gaussian Naive Bayes shows a very high performance with no concept-drift.

set of neighboring nodes. The computational cost to calculate the Markov blanket of a feature is prohibitive when dealing with high dimensional data.

Smart Preprocessing for Streaming Data (SPSD) is an approach that uses min-max normalization of numerical features [30]. The authors use two metrics to avoid unnecessary renormalization. SPSD only renormalizes when a threshold exceeds some threshold value of the metrics. Streaming data joins equal size chunk where all operations originate. The first data chunk is used to take the references min-max values and to send the normalized data for the training model. The metric 1 represents the amount of sample falling outside the min-max reference values; the metric 2 is the relation between new sample values in each dimension and the referenced min-max value for that dimension. Similar to our proposal, the algorithm works with numerical data.

Incremental Discretization Algorithm (IDA) uses a quantile approach to discretize data stream [26]. The algorithm discretizes data stream in m equal frequency bins. A sliding window version of the algorithm is proposed to follow the evolution of the data stream. The algorithm maintains the data into bins with fixed quantiles of the distribution, rather than fixed absolute values, to follow the distribution drift.

In our proposal, we propose an unsupervised preprocessing method. Our method includes normalization, and feature selection altogether. The proposal is parametric-less. Our algorithm follows an active approach for concept-drift detection. The active approach monitors the concept, the label, to determine when drift occurs before taking any action. A passive approach, in contrast, updates the model every time new data arrives, wasting resources. We modified our proposed Feature Selection algorithm to calculate the correlation between features in a sliding window. Also, a normalization algorithm is proposed to handle data stream.

VI. Conclusion

Achieving good classification metrics for streaming data is a challenge because neither the number of samples nor the domain of each sample feature is bounded. Therefore, it is mandatory to apply preprocessing methods to streaming data to bound the domain of each feature and to select only the most representative features for the classification model. In this paper, we presented a method for data preprocessing for classification of network traffic. The method is composed of two algorithms. First, we propose a normalization algorithm that enforces data to a normal distribution within values between -1 and 1, which produces a more accurate classification model and reduces the classification error. Then, a feature selection algorithm calculates the correlation of all pairs of features and selects the best features in an unsupervised way.

We select the features with the highest absolute correlation. When compared with traditional feature selection algorithms, our proposal selects an optimized subset of features improving the accuracy by more than 11% within a 100-fold reduction in processing time. Moreover, we modified our preprocessing method to work both on batch and on streaming data. When applied over streaming data, our preprocessing method was able to detect concept-drift due to the sensibility for detecting changes on the accuracy over a threshold.

ACKNOWLEDGMENT

The authors would like to thank Antonio Lobato, Igor Alvarenga and Igor Sanz for their significant contributions to obtain the results. This research is supported by CNPq, CAPES, FAPERJ, and FAPESP (2015/24514-9, 2015/24485-9, and 2014/50937-1).

REFERENCES

[1] Hu, P., Li, H., Fu, H., Cansever, D., and Mohapatra, P., "Dynamic defense strategy against advanced persistent threat with insiders," in *IEEE Conference on Computer Communications (INFOCOM)*, 4 2015, pp. 747–755.

[2] Paxson, V., "Bro: a System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[3] Roesch, M., "Snort-Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th USENIX conference on System administration*. USENIX Association, 1999, pp. 229–238.

[4] Vallentin, M., Sommer, R., Lee, J., Leres, C., Paxson, V., and Tierney, B., "The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware," in *Recent Advances in Intrusion Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 107–126.

[5] Bar, A., Finamore, A., Casas, P., Golab, L., and Mellia, M., "Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis," in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 10 2014, pp. 165–170.

[6] Stonebraker, M., Çetintemel, U., and Zdonik, S., "The 8 requirements of real-time stream processing," *ACM SIGMOD Record*, vol. 34, no. 4, pp. 42–47, 2005.

[7] Mayhew, M., Atighetchi, M., Adler, A., and Greenstadt, R., "Use of machine learning in big data analytics for insider threat detection," in *IEEE Military Communications Conference, MILCOM*, 10 2015, pp. 915–922.

[8] Mladenić, D., "Feature Selection for Dimensionality Reduction," in *Subspace, Latent Structure and Feature Selection (SLSFS): Statistical and Optimization Perspectives Workshop.*, Saunders, C., Grobelnik, M., Gunn, S., and Shawe-Taylor, J., Eds. Bohinj, Slovenia: Springer Berlin Heidelberg, 2006, pp. 84–102.

[9] Bifet, A. and Morales, G. D. F., "Big data stream learning with samoa," in *2014 IEEE International Conference on Data Mining Workshop*, Dec 2014, pp. 1199–1202.

[10] Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., and Ghédira, K., "Discussion and review on evolving data streams and concept drift adapting," *Evolving systems*, vol. 9, no. 1, pp. 1–23, 2018.

[11] Rahm, E. and Do, H. H., "Data cleaning: Problems and current approaches," *IEEE Bulletin of the Technical Committee on Data Engineering*, vol. 23, no. 4, pp. 3–13, 2000.

[12] García, S., Luengo, J., and Herrera, F., *Data preprocessing in data mining*. Springer, 2016.

[13] Robnik-Šikonja, M. and Kononenko, I., "Theoretical and Empirical Analysis of ReliefF and RReliefF," *Machine Learning*, vol. 53, no. 1/2, pp. 23–69, 2003.

[14] Schölkopf, B., Smola, A. J., and Müller, K.-R., "Kernel principal component analysis," in *Advances in kernel methods*. MIT Press, 1999, pp. 327–352.

[15] García, S., Luengo, J., and Herrera, F., "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining," *Knowledge-Based Systems*, vol. 98, pp. 1–29, 4 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0950705115004785

[16] Zhang, S., Zhang, C., and Yang, Q., "Data preparation for data mining," *Applied artificial intelligence*, vol. 17, no. 5-6, pp. 375–381, 2003.

[17] Tan, S., "Neighbor-weighted k-nearest neighbor for unbalanced text corpus," *Expert Systems with Applications*, vol. 28, no. 4, pp. 667–671, 2005.

[18] Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., and Herrera, F., "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, 2017.

[19] Van Der Maaten, L., Postma, E., and den Herik, J., "Dimensionality reduction: a comparative," *Journal of Machine Learning Research*, vol. 10, pp. 66–71, 2009.

[20] Ang, J. C., Mirzal, A., Haron, H., and Hamed, H. N. A., "Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 5, pp. 971–989, 9 2016.

[21] Chandrashekar, G. and Sahin, F., "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[22] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V., "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[23] Hall, M. A., "Correlation-based Feature Selection for Machine Learning," Ph.D. dissertation, The University of Waikato, 1999.

[24] Kumar, A., Sung, M., Xu, J. J., and Wang, J., "Data streaming algorithms for efficient and accurate estimation of flow size distribution," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1. ACM, 2004, pp. 177–188.

[25] Ben-Haim, Y. and Tom-Tov, E., "A streaming parallel decision tree algorithm," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 849–872, 2010.

[26] Webb, G. I., "Contrary to popular belief incremental discretization can be sound, computationally efficient and extremely useful for streaming data," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2014, pp. 1031–1036.

[27] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A., "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, July 2009, pp. 1–6.

[28] Lobato, A., Andreoni Lopez, M., Sanz, I. J., Cárdenas, A., Duarte, O. C. M. B., and Pujolle, G., "An adaptive Real-Time architecture for Zero-Day threat detection," in *IEEE ICC 2018 Next Generation Networking and Internet Symposium (ICC'18 NGNI)*, Kansas City, USA, May 2018.

[29] Andreoni Lopez, M., Silva, R. S., Alvarenga, I. D., Rebello, G. A. F., Sanz, I. J., Lobato, A. G. P., Mattos, D. M. F., Duarte, O. C. M. B., and Pujolle, G., "Collecting and characterizing a real broadband access network traffic dataset," in *IEEE/IFIP 1st Cyber Security in Networking Conference (CSNet)*, Oct 2017, pp. 1–8.

[30] Hu, H. and Kantardzic, M., "Smart preprocessing improves data stream mining," in *49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2016, pp. 1749–1757.

[31] Buczak, A. and Guven, E., "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys Tutorials*, no. 99, pp. 1–26, 2015.

[32] Prasath, V. B. S., Alfeilat, H. A. A., Lasassmeh, O., and Hassanat, A. B. A., "Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier - {A} Review," *CoRR*, vol. abs/1708.0, 2017. [Online]. Available: http://arxiv.org/abs/1708.04321

[33] Zhang, T., "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 116.

[34] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[35] Perkins, S. and Theiler, J., "Online feature selection using grafting," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 592–599.

[36] Zhou, J., Foster, D. P., Stine, R. A., and Ungar, L. H., "Streamwise feature selection," *Journal of Machine Learning Research*, vol. 7, no. Sep, pp. 1861–1885, 2006.

[37] Wu, X., Yu, K., Ding, W., Wang, H., and Zhu, X., "Online feature selection with streaming features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 5, pp. 1178–1192, 2013.