Machine Learning Methods for Spamdexing Detection

Renato M. Silva*, Tiago A. Almeida**, Akebo Yamakami*

*School of Electrical and Computer Engineering, University of Campinas – UNICAMP.
Campinas, São Paulo, Brazil. e-mail: {renatoms, akebo}@dt.fee.unicamp.br
**Department of Computer Science, Federal University of São Carlos – UFSCar.
Sorocaba, São Paulo, Brazil. e-mail: talmeida@ufscar.br

Abstract—In this paper, we present recent contributions for the battle against one of the main problems faced by search engines: the spamdexing or web spamming. They are malicious techniques used in web pages with the purpose of circumvent the search engines in order to achieve good visibility in search results. To better understand the problem and finding the best setup and methods to avoid such virtual plague, in this paper we present a comprehensive performance evaluation of several established machine learning techniques. In our experiments, we employed two real, public and large datasets: the WEBSPAM-UK2006 and the WEBSPAM-UK2007 collections. The samples are represented by content-based, link-based, transformed link-based features and their combinations. The found results indicate that bagging of decision trees, multilayer perceptron neural networks, random forest and adaptive boosting of decision trees are promising in the task of web spam classification.

Keywords—Spamdexing; web spam; spam host; classification, WEBSPAM-UK2006, WEBSPAM-UK2007.

1. Introduction

The web is growing by leaps and bounds and becoming an increasingly important source of entertainment, communication, research, news and trade. More users are having access to the Internet and the time they remain connected is also increasing. As a consequence, the competition between websites is highly motivated since there is a great interest in keep up in a good position in search results.

The desirable consequence of such competition is the quality improvement of the services provided. However, one of the bad resul is the emerging virtual plague known as web spam or spamdexing, which are web pages that employ techniques to circumvent the search engines to achieve better positions in search results [1], [2].

Web spam can provoke several problems. For instance, it can degrade the quality of search results and the nuisance to users by forging undeserved and unexpected answers, by promoting the announcement of unwanted pages [3]. It can also increase the computational cost of query processing and search engine indexing process. Furthermore, it can expose users to malicious content that installs malwares on their computers and can steal sensitive information, as passwords, financial information, or web-banking credentials, or degrade the performance of computers and network [4].

Recent studies indicate that the amount of web spam is dramatically increasing. In a research started in August 2010, John *et. al* [5] observed that 36% of the query results of Google and Bing contain malicious URLs. Lu *et. al* [6] ranked the most popular query terms used in Google and Bing, between September 2010 and April 2011, and they found that, on average, 50% of them return results with malicious URLs.

A report produced by the company Websense[1] shows that 22.4% of the search results about entertainment present malicious links. Furthermore, a report published by McAfee[2] informs that 49% of the most popular search terms return some malicious site in the top 100 search results. The same study found that 1.2% of the queries return links of the malicious sites in the top 100 results. According to the Google Online Security Blog[3], Google detects about 9,500 new malicious web sites every day and in about 12 to 14 million queries every day.

Given this scenario, this paper presents a comprehensive performance evaluation of several well-known machine learning techniques employed to automatically detect web spam in order to provide good baseline results for further comparison. Separated pieces of this work were presented at ICAI 2012 [7], IBERAMIA 2012 [8] and IEEE ICMLA 2012 [9]. Here, we have connected all ideas in a very consistent way. We have also offered a lot more details about each study and extended the performance evaluation using the well-known WEBSPAM-UK2007 dataset.

This paper is organized as follows: in Section 2 we describe the problem focusing in content-based and link-based spam techniques. Section 3 presents the basic concepts of the methods evaluated in this paper. In Section 4 we present the database and settings that we have used in our experiments. Section 5 presents the main results. Finally, Section 6 describes the conclusions and guidelines for future work.

1. Websense 2010 Threat Report. See: http://www.websense.com/assets/reports/report-websense-2010-threat-report-en.pdf

2. McAfee Threats Report: First Quarter 2011. See: http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2011.pdf

3. Google Online Security Blog – Safe Browsing: Protecting Web Users for 5 Years and Counting. See: http://googleonlinesecurity.blogspot.com.br/2012/06/safe-browsing-protecting-web-users-for.html

2. Web spamming techniques

There are several different web spamming techniques, such as cloaking spam [1], [10], [11], redirection spam [1], [12] and click spam [13], [12]. However, in this paper we addressed only the two most popular web spamming techniques: the content-based spam [1], [12] and link-based spam [13], [3], [14], [1], [12].

2.1 Content-based spam

Some search engines analyze the textual content of the web pages in order to determine their relevance. In this process, they collect terms in different positions on the web page. This terms are used to categorize the web page in one or more groups of pages that address similar subjects and to determine the relevance of the web page with respect to a query term or group of query terms.

The content-based spam is a technique that manipulates the terms of the spam page textual content in order to get it categorized into groups of web pages in which the spammer has interest and, consequently, to achieve a relevance score that does not match the quality of its content.

According to Gyongyi and Garcia-Molina [1], a very simple example of content spam is a web page with pornography and thousands of invisible keywords that have no connection with the pornographic content. Thus, when a user performs a query using one of the terms presented in these keywords, the search engines may return this pornographic web page as a result.

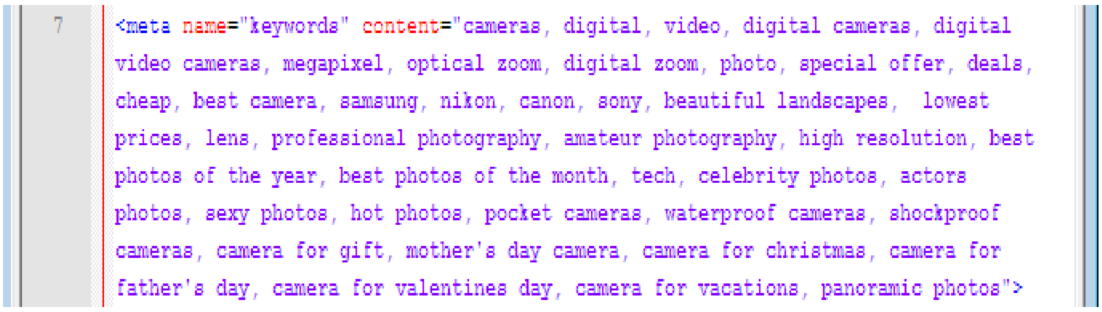
There are different types of content spam, such as [1], [12]:

- Title spam: this category of web spam includes popular keywords in the title of the web page. It is mainly because the most of search engines gives great weight to the terms presented in the document title.
- Meta tag spam: the *meta tags* are HTML code that describes the web page content or provide keywords, authors name or other information that are not visible to users, but can be examined by search engines. The spammers add keywords in meta tags because some search engines give some relevance to them. An example of meta tag spam is presented in Figure 1.
- Anchor text spam: the anchor text is used to summarize through keywords the subject addressed by the target document of a link. As well as in the title spam technique, the spammers also include keywords that do not correspond to the real content of the target documents of the links.
- URL spam: some search engines break the URL of a web page in a series of terms to determine its relevancy. So, for instance, if a spammer wants to increase the relevance of a web page of the your website for the term "cheap smartphones", it can create the following URL: *cheap-smartphones.com/cheap/smartphones.html*.
- Alt text spam: the alt tag is used to describe an image in a web page. This description is seen by the user only when the image is not loaded by the browser. However, it is always seen by the search engines and some of them give some relevance to the terms presented in this tag. Some spammers add images on web pages and insert popular keywords in alt tags, without worrying whether they are related to the image that they represent. Often, spammers include many images with minimum sizes or they make them invisible, because the only goal is to achieve scores with the keywords presented in the alt tags. A simple example of the alt text spam: *<img src="cheap-smartphones.com/cheap/smartphones.jpg" alt="cheap smartphone, smartphone sales, phone, mobile phone, cellphone, wifi, Internet, social networking, best smartphone, best android smartphone, samsung, apple, galaxy, iphone, lumia, ios" />*.
- Body spam: this technique is one of the simplest and most commonly used by spammers. It consists in including terms in the body of the web page. Some spammers hide these terms through of web programming scripts. One of the basic strategies is to make the text the same color as of the web page background or put it inside CSS (Cascading Style Sheets) layers and thereafter, make this layers invisible. Therefore, these terms are "seen" by the search engines, but only the content of interest of the spammer is presented to the user.

2.2 Link-based spam

Link-based spam is a technique that manipulates the page link structure in order to increase its score of relevance. This is because the one important criteria that search engines determines the relevance of a web page is by analyzing the amount of incoming links [13], [3], [14].

One of the main strategies used by the spammers is to create one or several web sites with thousands of links pointing to a web page of interest. Thus, the search engines can be deceived and give high



```
<meta name="keywords" content="cameras, digital, video, digital cameras, digital
video cameras, megapixel, optical zoom, digital zoom, photo, special offer, deals,
cheap, best camera, samsung, nikon, canon, sony, beautiful landscapes,  lowest
prices, lens, professional photography, amateur photography, high resolution, best
photos of the year, best photos of the month, tech, celebrity photos, actors
photos, sexy photos, hot photos, pocket cameras, waterproof cameras, shockproof
cameras, camera for gift, mother's day camera, camera for christmas, camera for
father's day, camera for valentines day, camera for vacations, panoramic photos">
```

Fig. 1: Example of a meta tag spam.

relevance ranking for this web page due to false popularity created by the web spamming technique. The main problem in detecting this kind of web spam is that the web page of interest is often reputable web pages that sells products or transmit messages or ideas [14], [1], [12]. Figure 2 presents an example of a link spam.

Note that, Figure 2a presents a web page without trace of spam. However, if we observe its source code, shown in Figure 2b, we can see that the web page is a link spam because it has several links to other pages with the single purpose of increasing their relevance ranking. One of the parts highlighted in Figure 2b presents the values “-1919px” and “-2932px”. Such numbers determines the position of the layer where the links are contained. Since the values are negative, the links that are in this layer are not seen by the user, but are “seen” by web crawlers. We can notice that the link highlighted in Figure 2b points to URL of the web page shown in Figure 2c. As can be observed, this site has commercial purposes and for this reason there is an interest in increasing its position in query results. The unethical nature of this site is clearly visible since it sells medicines without prescription and still illegally uses the Google company logo.

Another strategy used by spammers is to add links in several forums, discussion groups and comments in blogs and sites that points to the web page of interest. As a consequence, such web page appears to have a great popularity, since reliable web sites have links pointing to it [1], [12].

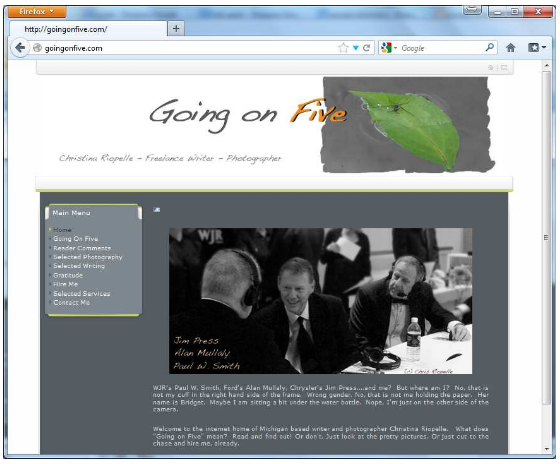
3. Methods

This section presents the main concepts regarding the following well-known methods that we have evaluated in this paper: multilayer perceptron neural networks, support vector machines, methods based on trees, such as decision trees, random forest, bagging and adaptive boosting of trees, and k-nearest neighbor. Such methods were chosen because most of them have been evaluated and presented as the best machine learning and data mining techniques currently available [15].

3.1 Multilayer perceptron neural network (MLP)

A multilayer perceptron neural network is a perceptron-type network that has a set of sensory units composed by an input layer, one or more intermediate (hidden) layers, and an output layer of neurons [16]. By default, MLP is a supervised learning method that uses the backpropagation algorithm which can be summarized in two stages: forward and backward [17].

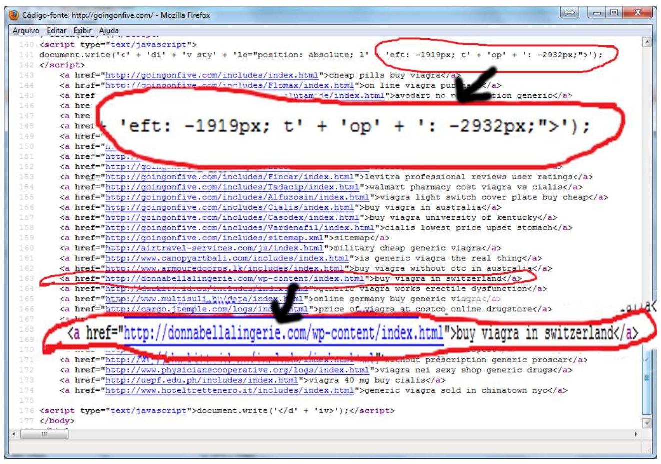
In the forward stage, the signal propagates through the network, layer by layer, as follows:



WJR's Paul W. Smith, Ford's Alan Mullaly, Chrysler's Jim Press....and me? But where am I? No, that is not my cuff in the right hand side of the frame. Wrong gender. No, that is not me holding the paper. Her name is Bridget. Maybe I am sitting a bit under the water bottle. Nope, I'm just on the other side of the camera.

Welcome to the internet home of Michigan based writer and photographer Christina Riopelle. What does "Going on Five" mean? Read and find out! Or don't. Just look at the pretty pictures. Or just cut to the chase and hire me, already.

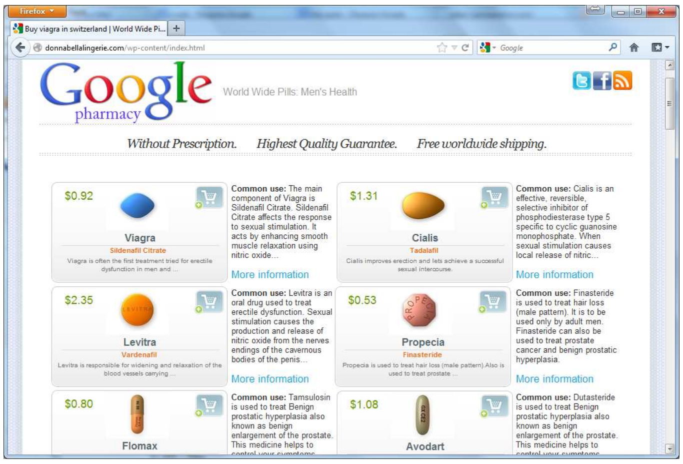
(a) Apparently ordinary web page.


Código-fonte: http://goingonfive.com/ - Mozilla Firefox
Arquivo Editar Exibir Ajuda
'eft: -1919px; t' + 'op' + ': -2932px;">');
<a href="http://donnabellalingerie.com/wp-content/index.html">buy viagra in switzerland</a>

```
<script type="text/javascript">
document.write('<' + 'di' + 'v sty' + 'le="position: absolute; l' + 'eft: -1919px; t' + 'op' + ': -2932px;">');
</script>
    <a href="http://goingonfive.com/includes/index.html">cheap pills buy viagra</a>
    <a href="http://goingonfive.com/includes/Flomax/index.html">on line viagra pur[illegible]</a>
    <a href="[illegible]lutam[illegible]de/index.html">avodart no [illegible]tion generic</a>
    <a hre[illegible]
    <a hre[illegible]
    <a hre[illegible]
    <a href="[illegible]
    <a href="[illegible]
    <a href="http:[illegible]
    <a href="http://goi[illegible]
    <a href="http://goingonfive.com/includes/Fincar/index.html">levitra professional reviews user ratings</a>
    <a href="http://goingonfive.com/includes/Tadacip/index.html">walmart pharmacy cost viagra vs cialis</a>
    <a href="http://goingonfive.com/includes/Alfuzosin/index.html">viagra light switch cover plate buy cheap</a>
    <a href="http://goingonfive.com/includes/Cialis/index.html">buy viagra in australia</a>
    <a href="http://goingonfive.com/includes/Casodex/index.html">buy viagra university of kentucky</a>
    <a href="http://goingonfive.com/includes/Vardenafil/index.html">cialis lowest price upset stomach</a>
    <a href="http://goingonfive.com/includes/sitemap.xml">sitemap</a>
    <a href="http://airtravel-services.com/js/index.html">military cheap generic viagra</a>
    <a href="http://www.canopyartbali.com/includes/index.html">is generic viagra the real thing</a>
    <a href="http://www.armouredcorps.lk/includes/index.html">buy viagra without otc in australia</a>
    <a href="http://donnabellalingerie.com/wp-content/index.html">buy viagra in switzerland</a>
    <a href="http://[illegible]">generic viagra works erectile dysfunction</a>
    <a href="http://www.multisuli.hu/data/index.html">online germany buy generic viagra</a>
    <a href="http://cargo.jtemple.com/logs/ind[illegible]html">price of viagra at costco online drugstore</a>
    [illegible]
    <a href="http://[illegible]">[illegible]without prescription generic proscar</a>
    <a href="http://www.physicianscooperative.org/logs/index.html">viagra nei sexy shop generic drugs</a>
    <a href="http://uspf.edu.ph/includes/index.html">viagra 40 mg buy cialis</a>
    <a href="http://www.hoteltrettenero.it/includes/index.html">generic viagra sold in chinatown nyc</a>

<script type="text/javascript">document.write('</d' + 'iv>');</script>
</body>
```

(b) Web page source code.



Viagra is often the first treatment tried for erectile dysfunction in men and ...

Common use: The main component of Viagra is Sildenafil Citrate. Sildenafil Citrate affects the response to sexual stimulation. It acts by enhancing smooth muscle relaxation using nitric oxide...

Cialis improves erection and lets achieve a successful sexual intercourse.

Common use: Cialis is an effective, reversible, selective inhibitor of phosphodiesterase type 5 specific to cyclic guanosine monophosphate. When sexual stimulation causes local release of nitric...

Levitra is responsible for widening and relaxation of the blood vessels carrying ...

Common use: Levitra is an oral drug used to treat erectile dysfunction. Sexual stimulation causes the production and release of nitric oxide from the nerves endings of the cavernous bodies of the penis...

Propecia is used to treat hair loss (male pattern).Also is used to treat prostate ...

Common use: Finasteride is used to treat hair loss (male pattern). It is to be used only by adult men. Finasteride can also be used to treat prostate cancer and benign prostatic hyperplasia.

Common use: Tamsulosin is used to treat Benign prostatic hyperplasia also known as benign enlargement of the prostate. This medicine helps to control your symptoms

Common use: Dutasteride is used to treat Benign prostatic hyperplasia also known as benign enlargement of the prostate. This medicine helps to control your symptoms

(c) Web page that the spammer intends to promote.

Fig. 2: Example of link spam.

$u_j^l(n) = \sum_{i=0}^{m^{l-1}} w_{ji}^l(n) y_i^{l-1}(n)$, where $l = 0, 1, 2, ..., L$ are the indexes of network layers. So, $l = 0$ represents the input layer and $l = L$ represents the output layer. On the other hand, $y_i^{l-1}(n)$ is the output function relating to the neuron i in the previous layer, $l-1$, $w_{ji}^l(n)$ is the synaptic weight of neuron j in layer l and m^l corresponds to the number of neurons in layer l. For $i = 0$, $y_0^{l-1}(n) = +1$ and $w_{j0}^l(n)$ represent the bias applied to neuron j in layer l [16].

The output of neuron j in layer l is given by $y_j^l(n) = \varphi_j(u_j^l(n))$, where φ_j is the activation function of j. Then, the error can be calculated by $e_j^l(n) = y_j^l(n) - d(n)$, where $d(n)$ is the desired output for an input pattern $x(n)$.

In backward stage, the derivation of the backpropagation algorithm is performed starting from the output layer, as follows: $\delta_j^L(n) = \varphi_j'(u_j^L(n)) e_j^L(n)$, where φ_j' is the derivative of the activation function. For $l = L, L-1, ..., 2$, is calculated: $\delta_j^{l-1}(n) = \varphi_j'(u_j^{l-1}(n)) \sum_{i=1}^{m^l} w_{ji}^l(n) * \delta_j^l(n)$, for $j = 0, 1, ..., m^l - 1$.

Consult Haykin [16] and Bishop [17] for more details.

3.1.1 Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm is usually employed to optimize and accelerate the convergence of the backpropagation algorithm [17]. It is considered a second order method because it uses information about the second derivative of the error function. Details can be found in Bishop [17] and Hagan and Menhaj [18].

3.2 Kohonen's self-organizing map

The Kohonen's self-organizing map (SOM) is based on unsupervised competitive learning. Its main purpose is to transform an input pattern of arbitrary dimension in a one-dimensional or two-dimensional map in a topologically ordered fashion [16], [19].

The training algorithm for a SOM can be summarized in two stages: competition and cooperation [16], [19].

In the competition stage, a random input pattern (x_j) is chosen, the similarity between this pattern and all the neurons of the network is calculated by the Euclidean distance $id = \arg \underset{\forall i}{min} \|x_j - w_i\|$ where $i = 1, ...k$, and the index of the neuron with lowest distance is selected.

In cooperation stage, the synaptic weights w_{id} that connect the winner neuron in the input pattern x_i is updated. The weights of neurons neighboring the winner neuron are also updated by $w_i(t+1) = w_i(t) + \alpha(t)h(t)(x_i - w_i(t))$, where t is the number of training iterations, $w_i(t+1)$ is the new weight vector, $w_i(t)$ is the current weight vector, α is the learning rate, $h(t)$ is the neighborhood function and x_i is the input pattern.

The neighborhood function $h(t)$ is equal to 1 when the winner neuron is updated. This is because it determines the topological neighborhood around the winning neuron, defined by the neighborhood radius σ. The amplitude of this neighborhood function monotonically decreases as the lateral distance between the neighboring neuron and the winner neuron increases. There are several ways to calculate this neighborhood function, and one of the most common is the Gaussian function, defined by $h_{ji}(t) = \exp\left(\frac{-d_{ji}^2}{2\sigma^2(t)}\right)$, where d_{ji} is the lateral distance between winner neuron i and neuron j. The parameter $\sigma(t)$ defines the neighborhood radius and should be some monotonic function that decreases over the time. So, the exponential decay function $\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau}\right)$ can be used, where σ_0 is the initial value of σ, t is the current iteration number and τ is a time constant of the SOM, defined by $\tau = \frac{1000}{\log \sigma_0}$

The competition and cooperation stages are carried out for all the input patterns. Then, the neighborhood radius σ and learning rate α are updated. This parameter should decrease with time and can be calculated by $\alpha(t) = \alpha_0 \exp\left(-\frac{t}{\tau}\right)$, where α_0 is the initial value of α, t is the current iteration number and τ is a time constant of the SOM which can be calculated as presented in the cooperation stage.

3.3 Learning vector quantization

The learning vector quantization (LVQ) is a supervised learning technique that aims to improve the quality of the classifier decision regions, by adjusting the feature map through the use of information about the classes [16].

According to Kohonen [19], the SOM can be used to initialize the feature map by defining the set of weight vectors w_{ij}. The next step is to assign labels to neurons. This assignment can be made by majority vote, in other words, each neuron receives the class label in that it is more activated.

After this initial step, the LVQ algorithm can be employed. Although, the training process is similar to the SOM one, it does not use neighborly relations. Therefore, it is checked if the class of the winner neuron is equal to the class of the input vector x, and it is updated as follows:

$$w_{id}(t+1) = \begin{cases} w_{id}(t) + \alpha(t)(x_i - w_{id}(t)), \text{ same class} \\ w_{id}(t) - \alpha(t)(x_i - w_{id}(t)), \text{ different class} \end{cases}$$

where α is the learning rate, id is the index of the winner neuron and t is the current iteration number.

3.4 Radial basis function neural network

A radial basis function neural network (RBF), in its most basic form, has three layers. The first one is the input layer which has sensory units connecting the network to its environment. The second layer is hidden and composed by a set of neurons that use radial basis functions to group the input patterns in clusters. The third layer is the output one, which is linear and provides a network response to the activation function applied to the input layer [16]. The activation function most common for the RBFs is the Gaussian, defined by $h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$, where x is the input vector, c is the center point and r is the width of the function.

The procedure for training a RBF is performed in two stages. In the first one, the parameters of the basic functions related to the hidden layer are determined through some method of unsupervised training, as K-means.

In the second training phase, the weights of the output layer are adjusted, which corresponds to solve a linear problem [17]. According to Bishop [17], considering an input vector $x = [x_1, x_2, ..., x_n]$, the network output is calculated by $y_k = \sum_{j=1}^{m} w_{kj} h_j$, where $x = [w_{k1}, w_{k2}, ..., x_{km}]$ are the weights, $h = [h_1, h_2, ..., h_m]$ are the radial basis functions, calculated by a function of radial basis activation.

After calculating the outputs, the weights should be updated. A formal solution to calculate the weights is given by $w = h^{\dagger}d$, where h is the matrix of basis functions, $h^{\dagger}$ represents the pseudo-inverse of h and d is a vector with the desired responses [17].

Consult Haykin [16], Bishop [17] and Orr [20] for more information.

3.5 Support vector machines (SVM)

Support vector machines (SVM) [21] is a machine learning method that can be used for pattern classification, regression and others learning tasks [16], [22]. This method was conceptually implemented following the idea that input vectors are non-linearly mapped to a high dimension feature space. In this feature space is constructed a linear decision surface which separates the classes of the input patterns.

One of the main elements that the SVM uses to separate the patterns of distinct classes is a kernel function. Through it, the SVM constructs a decision surface nonlinear in the input space, but linear in the features space [16]. Table 1 presents the most popular SVM kernel functions, where the γ parameter controls the shape of the decision surface, r controls the displacement threshold of the polynomial and sigmoid kernels and d is the degree of the polynomial kernel. The γ, r and d must be set by the user.

To assist the choice of the SVM parameters, Hsu *et. al* [23] recommend the employment of a grid search. For instance, considering the SVM with RBF kernel, in which is necessary to define the regularization parameter C and γ, the authors suggest that the grid search could be used to evaluate

TABLE 1: The most popular SVM kernel functions [16], [23].

Linear	$k(x_i, x_j) = x_i^T x_j$
RBF	$k(x_i, x_j) = \exp(-\frac{1}{2\gamma^2}\|x_i - x_j\|^2), \gamma > 0$
Polynomial	$k(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
Sigmoid	$k(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

exponential sequences: $C = 2^{-5}, 2^{-4}, 2^{-3}..., 2^{15}$ and $\gamma = 2^{-15}, 2^{-14}..., 2^{3}$.

3.6 Decision trees (C4.5)

The C4.5 [24] is one of the most classical decision tree algorithms and uses both categorical and continuous attributes. It uses a divide-and-conquer approach to increase the predictive ability of the decision trees. Thus, a problem is divided in several sub-problems, by creating sub-trees in the path between the root and the leaves of the decision tree.

3.7 Random forest

A random forest [25] is a combination of decision trees in which each tree depends on the values of a random vector sampled independently and equally distributed for all trees in the forest. In this method, after generating a large number of trees, each one votes for one class of the problem. Then, the class with more number of votes is chosen.

3.8 K-nearest neighbors (IBK)

The IBK is an instance-based learning algorithm (IBL) [26]. Such method, derived from the k-nearest neighbors (KNN) classifier, is a non-incremental algorithm and aims to keep a perfect consistency with the initial training set. On the other hand, the IBL algorithm is incremental and one of its goals is maximizing classification accuracy on new instances [26].

As well as in the KNN, in IBK, the classification generated for the sample i is influenced by the outcome of the classification of its k-nearest neighbors, because similar samples often have similar classifications [26], [27].

3.9 Adaptive boosting (AdaBoost)

The adaptive boosting [28] is a boosting algorithm widely used in pattern classification problems. In general, as any boosting method, it makes a combination of classifiers. However, it has some properties that make it more practical and easier to implement than the boosting algorithms that preceded it. One of these properties is that it does not require any prior knowledge of the predictions achieved by weak classifiers. Instead, it adapts to the bad predictions and generates a weighted majority hypothesis in which the weight of the each prediction achieved by weak classifiers it is a function of its prediction.

3.10 Bagging

The bagging [29] is a method for generating multiple versions of a classifier that are combined to achieve an aggregate classifier. The classification process is similar to the boosting methods, but according to Witten *et. al* [27], unlike what occurs in the second one, in the bagging, the different models of classifiers get the same weight in the generation of a prediction.

3.11 LogitBoost

The LogitBoost method [30] is a statistical version of the boosting method and, according to Witten *et. al* [27], it has some similarities with Adaboost, but it optimizes the likelihood of a class, while the Adaboost optimizes an exponential cost function.

Friedman *et. al* [30] defines this method as an algorithm for assembly of additive logistic regression models.

3.12 OneR

The OneR method or 1R (1-rules) [31] can be considered a 1-level decision tree because it generates a set of rules, one for each feature of the dataset, and classifies a sample based on a single feature. The chosen is one whose rule produces the smallest error rates.

3.13 Naive Bayes

The naive Bayes method [32] is a simple probabilistic classifier based on Bayes theorem. This method is termed "naive" because it assumes that the features contribute independently to the probability of occurrence of a class. Therefore, according to Witten *et. al* [27] the redundant features can skew the learning process.

4. Database and Experiment Settings

To give credibility to the found results and in order to make the experiments reproducible, all the tests were performed with two large, public and well-known datasets: the WEBSPAM-UK2006 and WEBSPAM-UK2007 collections[4]. The first collection is composed by 77.9 million web pages in 11,000 hosts in the UK domains. The second one is composed by 105,896,555 web pages in 114,529 hosts. Both of them were used in *Web Spam Challenge*[5], that is a well-known competition of web spam detection techniques.

4. Yahoo! Research: "Web Spam Collections". Available at http://barcelona.research.yahoo.net/webspam/datasets/.

5. *Web Spam Challenge:* http://webspam.lip6.fr/

In our experiments, we followed the same competition guidelines. In this way, three sets of features were employed to discriminate the hosts as spam or ham: the first one is composed by 96 content-based features [33], the second one is composed by 41 link-based features [34] and the third one is composed by 138 transformed link-based features [33], which are the simple combination or logarithm operation of the link-based features.

We have first preprocessed the data by removing all feature vectors with no label or labeled as undefined. After this, the set of features vectors extracted from WEBSPAM-UK2006 collection stayed with 6,509 (76.6%) hosts labeled as ham and 1,978 (23.4%) as spam. Further, the set of feature vectors extracted from WEBSPAM-UK2007 collection stayed with 5,476 (94.5%) hosts labeled as ham and 321 (5.5%) as spam.

To address the algorithms performance, we used a random sub-sampling validation, which is also known as Monte Carlo cross-validation [35]. Such method provides more freedom to define the size of training and testing subsets. Unlike the traditional k-fold cross-validation, the random sub-sampling validation allows to do as many repetitions were desired, using any percentage of data for training and testing.

Despite the Web Spam Challenge has provided pre-defined train and test split sets, we have not used them because we wanted to do at least ten experiments with each classifier in order to make a more consistent statistical analysis. With the original partition provided by the competition it would be possible to make no more than five experiments with each method. In this way, we divided each simulation in 10 tests in which we randomly selected 80% of the samples of each class to be presented to the algorithms in the training stage and the remaining ones were separated for testing. At the end, we calculated the arithmetic mean and standard deviation of the following well-known measures [27]:

- Accuracy: the overall hit rate, in others words, the proportion of correct predictions. This measures is defined by: $\frac{TP+TN}{TP+TN+FP+FN}$, where TP (true positives) refers to the number of examples correctly classified as spam, FP (false positives) refers to the examples that were incorrectly classified as spam, TN (true negatives) refers to the number of examples correctly classified as ham, and FN (false negatives) refers to the number of examples incorrectly classified as ham.
- Recall: proportion of spam correctly identified. Indicates how good the classifier is to identify the positive class. It is defined by: $\frac{TP}{TP+FN}$.
- Specificity: is the proportion of ham correctly identified. Indicates how good the classifier is to identify the negative class. It is defined by: $\frac{TP}{TN+FP}$.
- Precision: is the percentage of patterns classified as belonging to positive class and that really belong to positive class. It is defined by: $\frac{TP}{TP+FP}$.
- F-measure: is the harmonic mean between precision and recall. This measures is defined by: $2 * \frac{precision*recall}{precision+recall}$.

4.1 Settings

In the following, we describe the main parameters we have used for each classifier.

4.1.1 MLP

We evaluated the following well-known artificial neural networks: multilayer perceptron trained with the gradient descent method (MLP-GD) and multilayer perceptron trained with Levenberg-Marquardt method (MLP-LM).

We have implemented all the MLPs with a single hidden layer and with one neuron in the output layer. In addition, we have employed a linear activation function for the neuron of output layer and a hyperbolic tangent activation function for the neurons of the intermediate layer. Thus, we normalized the data for the interval $[-1, 1]$. Furthermore, one of the stopping criteria that we have used was the increasing of the validation set error (checked every 10 iterations). The others parameters were empirically calibrated and are the following:

- MLP-GD:
  - $\theta = 10,000$
  - $\gamma = 0.001$
  - Step learning $\alpha = 0.005$
  - Number of neurons in the hidden layer: 100
- MLP-LM:
  - $\theta = 500$
  - $\gamma = 0.001$
  - Step learning $\alpha = 0.001$
  - Number of neurons in the hidden layer: 50
- SOM + LVQ:
  - SOM step:
    - $\theta = 2,000$
    - Step learning $\alpha = 0.01$
    - Min step learning $\alpha_{min} = 0.01$
    - Number of neurons in the hidden layer: 150
    - Neighborhood function: one-dimensional
    - Initial neighborhood radius $\sigma = 4$
  - LVQ step:
    - $\theta = 2,000$
    - Step learning $\alpha = 0.01$
    - Number of neurons in the hidden layer: 120
- RBF:
  - Number of neurons in the hidden layer: 50

In the experiments with RBF neural network we have not employed any stopping criteria because the training is not iterative, since it employs the pseudo-inverse method [17]. Further, the dispersion of the each neuron of the RBF neural network was calculated by the following equation: $disp_i = \frac{1}{m}\sum_{j=1}^{m} dist(C_i, C_j)$, where m is the number of neurons or centers and $dist(C_i, C_j)$ is the Euclidean distance of the center C_i to center C_j.

4.1.2 SVM

We have implemented the SVM using the LIBSVM library [22] available for the MATLAB tool. We performed simulations with the linear, RBF, sigmoid and polynomial kernel functions and we have used grid search to define the parameters. However, in the SVMs with polynomial and sigmoid kernel, that have a larger number of parameters, we have performed the grid search only on the parameters C and γ, due to excessive computational cost. In this case, we have set the default values of the LIBSVM for the others parameters.

We have performed the grid search using the random sub-sampling validation with 80% of the samples for training and 20% for test. Then, we have chosen the best parameters – those that achieved the highest f-measures, and we have used them to perform the experiments. In such experiments we evaluated all types of mentioned kernels. However, due to space limit, we decided to present only the results achieved by SVM with RBF kernel since it achieved the best performance. Table 2 presents the parameters used in the simulations with the SVM.

4.1.3 Remaining methods

We have implemented the remaining classifiers using the WEKA tool [36]. The AdaBoost and bagging algorithms were trained with 100 iterations and both of them employ aggregation of multiple versions of C4.5 method. For all other approaches we have used the default parameters.

TABLE 2: Best parameters found by grid search and used by the SVM method with RBF kernel.

Types of feature vectors	C	γ
WEBSPAM-UK2006		
Unbalanced classes		
Content	2^{15}	2^{3}
Links	2^{15}	2^{-1}
Transformed links	2^{10}	2^{-9}
Content + links	2^{15}	2^{-2}
Links + transformed links	2^{15}	2^{0}
Content + transformed links	2^{15}	2^{3}
Content + links + transformed links	2^{15}	2^{-3}
Balanced classes		
Content	2^{14}	2^{3}
Links	2^{15}	2^{-1}
Transformed links	2^{5}	2^{-5}
Content + links	2^{14}	2^{-4}
Links + transformed links	2^{15}	2^{-3}
Content + transformed links	2^{15}	2^{3}
Content + links + transformed links	2^{15}	2^{-3}
WEBSPAM-UK2007		
Balanced classes		
Content	2^{10}	2^{3}
Links	2^{11}	2^{3}
Transformed links	2^{7}	2^{-1}
Content + links	2^{15}	2^{1}
Links + transformed links	2^{12}	2^{2}
Content + transformed links	2^{10}	2^{3}
Content + links + transformed links	2^{12}	2^{1}

5. Results

In this section we report the results achieved by the machine learning methods presented in Section 3. The experiments were performed with WEBSPAM-UK2006 and WEBSPAM-UK2007 datasets.

5.1 Results under WEBSPAM-UK2006

Regarding the experiments with WEBSPAM-UK2006 dataset, for each evaluated method and feature set, we have performed a simulation using unbalanced classes, as originally provided in the web spam competition, with 6,509 (76.6%) samples of ham hosts and 1,978 (23.4%) of spam ones. Moreover, to evaluate if the unbalance impacts on the classifiers performance, we have also performed simulations using the same number of instances in each class. For this, we have used the random undersampling [37], [38]. It balances the classes through a random elimination of samples belonging the majority class. After this, both classes were composed by 1,978 representatives.

Tables 3 and 4 present the results achieved by each classifier using content, links and transformed links-based features, respectively. In each table, the results are sorted by the F-measure achieved for each feature set. Each column shows the arithmetic means and the standard deviations of the results achieved using the random sub-sampling validation with 10 iterations. The bold values preceded by the symbol “↑” indicate the highest score and the bold values preceded by the symbol “↓” indicate the lowest score for each performance measure. Further, values preceded by the symbol “*” indicate the highest or lowest score considering the three set of features.

The results indicate that the evaluated classifiers are superior when are trained using balanced classes. Therefore, we have noted that the methods tend to be biased to the benefit of the class with the largest amount of samples. We can see that, in general, the specificity rate is higher than recall rate, which indicates that the classifiers have more successful to identify ham hosts since such class has more representatives.

The method used to balance the classes (random undersampling) did not affect the learning process of the algorithms. Otherwise, the standard deviations were expected to be much higher because the random undersampling method can select different instances to compose the test set at each repetition. Therefore, we can conclude that the balancing method is suitable for our purpose, because besides generating results with small standard deviations, it also greatly improved the performance of learning methods if compared to the accuracy achieved in the experiments with unbalanced classes.

Regarding the machine learning approaches, the results indicate that bagging of decision trees achieved the best overall performance. In average, it was able to detect 82.2% of the spam hosts with a precision rate of 82.7%. On the other hand, the naive Bayes and the neural network RBF achieved the worst results. However, is important to note that the naive Bayes method achieved satisfactory results for at least one of the features set – the link-based features.

With respect to the set of features, we can note that the best results were achieved when transformed link-based features were employed (Table 4).

5.1.1 Combinations of feature vectors extracted from WEBSPAM-UK2006 dataset.

One of the problems in spam host detection is that spammers generally employ more than one spamming technique. Often, spammers create web pages with both content and link spam because the search engines usually give a high score to web pages with a link structure that indicates that it is important for the web community with respect to one or more search queries at the same time that its content addresses such queries appropriately.

TABLE 3: Results achieved by each classifier using features extracted from WEBSPAM-UK2006 dataset with unbalanced classes.

		Accuracy	Recall	Specificity	Precision	F-measure
Content-based features	Bagging	↑**89.7** ± 0.5	68.7 ± 2.3	96.1 ± 0.6	*↑**84.4** ± 1.9	↑**0.757** ± 0.014
	Random forest	88.9 ± 1.0	65.7 ± 4.4	96.0 ± 1.2	83.4 ± 3.7	0.734 ± 0.024
	MLP-LM	88.6 ± 1.4	69.3 ± 4.2	94.2 ± 1.2	77.6 ± 4.6	0.731 ± 0.032
	AdaBoost	87.9 ± 0.9	66.6 ± 3.2	94.4 ± 0.7	78.4 ± 2.3	0.720 ± 0.024
	IBK	86.2 ± 0.8	64.6 ± 1.3	92.7 ± 0.7	72.8 ± 2.0	0.685 ± 0.011
	C4.5	85.1 ± 1.2	67.7 ± 4.6	90.4 ± 0.5	68.0 ± 0.6	0.678 ± 0.024
	MLP-GD	86.2 ± 1.2	57.0 ± 4.6	95.0 ± 0.4	77.5 ± 2.7	0.656 ± 0.039
	LogitBoost	84.3 ± 1.0	54.2 ± 3.9	93.5 ± 1.1	71.6 ± 3.2	0.616 ± 0.023
	SVM	82.5 ± 0.6	36.0 ± 2.4	*↑**96.6** ± 0.5	76.2 ± 2.8	0.488 ± 0.023
	OneR	80.8 ± 0.7	38.3 ± 2.4	93.7 ± 0.6	65.0 ± 2.7	0.482 ± 0.023
	SOM + LVQ	35.3 ± 3.7	94.7 ± 0.9	80.9 ± 0.7	67.0 ± 2.7	0.461 ± 0.033
	RBF	80.7 ± 0.6	↓**30.3** ± 2.7	96.1 ± 0.4	70.0 ± 2.2	0.422 ± 0.028
	Naive Bayes	↓**32.3** ± 4.6	*↑**97.4** ± 1.0	*↓**12.5** ± 6.2	*↓**25.4** ± 1.3	↓**0.402** ± 0.015
Link-based features	Bagging	↑**89.7** ± 0.6	79.2 ± 1.6	92.8 ± 0.9	↑**77.2** ± 2.0	↑**0.781** ± 0.009
	Random forest	89.1 ± 0.9	76.5 ± 7.9	92.8 ± 2.2	77.0 ± 3.7	0.764 ± 0.027
	AdaBoost	87.9 ± 0.8	72.6 ± 3.0	92.6 ± 0.8	74.9 ± 1.9	0.737 ± 0.019
	C4.5	87.2 ± 0.5	73.2 ± 2.1	91.5 ± 0.8	72.3 ± 1.4	0.727 ± 0.011
	MLP-LM	88.1 ± 1.6	70.8 ± 6.6	93.0 ± 0.7	74.1 ± 3.7	0.723 ± 0.049
	LogitBoost	86.7 ± 0.5	71.8 ± 4.0	91.2 ± 1.4	71.4 ± 2.1	0.715 ± 0.009
	MLP-GD	86.4 ± 1.3	61.6 ± 3.1	93.9 ± 0.9	75.3 ± 2.9	0.677 ± 0.026
	IBK	83.7 ± 1.4	67.8 ± 2.8	88.5 ± 1.3	64.2 ± 2.4	0.659 ± 0.023
	Naive Bayes	73.4 ± 1.2	↑**94.6** ± 1.3	↓**66.9** ± 1.5	↓**46.5** ± 1.2	0.624 ± 0.012
	SVM	81.2 ± 0.7	47.1 ± 2.5	91.5 ± 0.6	62.8 ± 1.9	0.538 ± 0.021
	OneR	78.6 ± 0.8	49.4 ± 3.5	87.5 ± 1.2	54.7 ± 1.9	0.518 ± 0.023
	SOM + LVQ	*↓**21.8** ± 3.6	94.3 ± 1.3	77.4 ± 0.6	54.0 ± 3.3	0.309 ± 0.035
	RBF	77.7 ± 0.5	*↓**17.2** ± 1.1	↑**96.1** ± 0.7	57.4 ± 4.3	*↓**0.264** ± 0.015
Transformed link-based features	Bagging	*↑**89.8** ± 0.7	78.9 ± 1.2	93.1 ± 0.8	↑**77.6** ± 2.1	*↑**0.782** ± 0.014
	Random forest	88.8 ± 0.4	76.5 ± 3.1	92.5 ± 1.5	75.8 ± 4.1	0.760 ± 0.012
	MLP-LM	89.0 ± 0.8	74.9 ± 3.9	93.1 ± 0.5	76.1 ± 2.1	0.754 ± 0.027
	SVM	88.3 ± 0.6	76.6 ± 2.3	91.8 ± 0.8	73.9 ± 1.6	0.752 ± 0.012
	MLP-GD	88.3 ± 0.9	75.2 ± 2.2	92.1 ± 1.4	73.9 ± 3.1	0.745 ± 0.014
	AdaBoost	87.9 ± 0.8	72.6 ± 3.0	92.6 ± 0.8	74.9 ± 1.9	0.737 ± 0.019
	LogitBoost	86.4 ± 0.4	70.2 ± 4.9	91.5 ± 1.5	72.3 ± 3.2	0.711 ± 0.019
	C4.5	86.3 ± 0.4	73.5 ± 1.8	90.0 ± 0.7	68.5 ± 2.2	0.709 ± 0.016
	SOM + LVQ	66.6 ± 2.6	92.2 ± 0.9	86.2 ± 0.6	72.2 ± 2.0	0.692 ± 0.015
	IBK	84.7 ± 0.7	67.5 ± 2.4	90.0 ± 1.5	67.3 ± 3.5	0.673 ± 0.011
	OneR	82.9 ± 1.0	↓**64.3** ± 2.1	88.6 ± 1.3	63.2 ± 2.8	0.637 ± 0.018
	RBF	84.7 ± 1.0	52.4 ± 6.2	↑**94.5** ± 1.1	74.3 ± 2.6	0.612 ± 0.043
	Naive Bayes	↓**35.9** ± 1.0	↑**96.5** ± 1.0	↓**17.5** ± 1.5	↓**26.2** ± 0.3	↓**0.412** ± 0.004

Therefore, this section presents the results achieved in experiments performed with all possible combinations of the feature vectors. The goal is to improve the results of the web spam classification by identifying hosts that employ content and link spam simultaneously.

Tables 5 and 6 present the results achieved by each method with the following features combinations: content and links, content and transformed links, links and transformed links, and content, links and transformed links. Bold values preceded by the symbol "↑" indicate the highest score and the bold values preceded by the symbol "↓" indicate the lowest score for each performance measure. Further, values preceded by the symbol "*" indicate the highest or lowest score considering all feature combinations.

The results indicate that using combinations of

TABLE 4: Results achieved by each classifier using features extracted from WEBSPAM-UK2006 dataset with balanced classes.

		Accuracy	Recall	Specificity	Precision	F-measure
Content-based features	MLP-LM	↑**87.6** ± 1.5	↑**86.5** ± 2.0	*↑**88.8** ± 1.9	*↑**89.1** ± 2.4	↑**0.877** ± 0.017
	Bagging	85.0 ± 1.2	83.5 ± 2.3	86.5 ± 1.5	86.1 ± 1.3	0.848 ± 0.013
	MLP-GD	84.7 ± 1.6	82.9 ± 2.0	86.4 ± 2.4	86.1 ± 2.4	0.845 ± 0.015
	AdaBoost	82.7 ± 1.3	81.8 ± 2.5	83.6 ± 1.9	83.3 ± 1.6	0.825 ± 0.014
	Random forest	82.8 ± 1.5	81.8 ± 3.3	83.8 ± 3.1	83.4 ± 2.3	0.825 ± 0.015
	IBK	79.9 ± 1.4	77.0 ± 2.6	82.7 ± 1.5	81.4 ± 1.5	0.791 ± 0.017
	C4.5	78.7 ± 0.4	79.2 ± 2.5	78.2 ± 1.9	78.3 ± 1.6	0.787 ± 0.012
	LogitBoost	78.3 ± 1.6	77.0 ± 3.2	79.7 ± 0.6	78.9 ± 1.1	0.779 ± 0.015
	Naive Bayes	↓**63.6** ± 7.5	83.6 ± 17.7	↓**43.6** ± 30.5	↓**62.8** ± 10.2	0.694 ± 0.039
	SOM + LVQ	65.6 ± 2.4	74.9 ± 2.5	70.3 ± 0.8	72.4 ± 1.5	0.688 ± 0.011
	SVM	71.1 ± 1.4	60.6 ± 2.7	81.6 ± 1.5	76.7 ± 1.5	0.677 ± 0.019
	OneR	66.5 ± 1.5	61.6 ± 1.0	71.4 ± 2.4	68.3 ± 2.0	0.648 ± 0.013
	RBF	68.9 ± 1.7	*↓**55.6** ± 1.9	82.3 ± 2.2	75.9 ± 2.6	*↓**0.641** ± 0.019
Link-based features	Bagging	↑**87.7** ± 1.3	91.7 ± 1.9	83.7 ± 1.7	84.9 ± 1.4	↑**0.882** ± 0.013
	Random forest	↑**87.7** ± 0.7	88.8 ± 2.5	↑**86.7** ± 2.1	↑**87.1** ± 1.6	0.879 ± 0.010
	MLP-LM	86.4 ± 1.7	92.1 ± 3.8	81.1 ± 2.6	82.3 ± 2.6	0.868 ± 0.019
	AdaBoost	85.5 ± 0.8	87.7 ± 1.7	83.2 ± 1.0	83.9 ± 0.8	0.858 ± 0.009
	MLP-GD	83.9 ± 1.9	90.2 ± 2.6	77.9 ± 3.0	80.0 ± 2.9	0.848 ± 0.019
	LogitBoost	83.2 ± 1.5	88.0 ± 3.7	78.4 ± 1.7	80.4 ± 1.0	0.840 ± 0.018
	C4.5	83.4 ± 0.9	85.4 ± 2.2	81.5 ± 1.7	82.3 ± 1.4	0.838 ± 0.011
	Naive Bayes	80.5 ± 1.8	↑**94.8** ± 0.9	↓**66.1** ± 3.3	73.7 ± 2.0	0.829 ± 0.014
	OneR	81.4 ± 1.2	88.7 ± 1.0	74.2 ± 2.2	77.5 ± 1.5	0.827 ± 0.010
	IBK	78.3 ± 0.4	80.8 ± 1.4	75.8 ± 1.6	77.1 ± 0.8	0.789 ± 0.003
	SVM	76.3 ± 1.4	77.1 ± 2.2	75.4 ± 2.4	75.8 ± 1.7	0.765 ± 0.014
	SOM + LVQ	70.1 ± 3.5	70.0 ± 2.8	70.0 ± 1.1	↓**70.0** ± 1.3	0.700 ± 0.017
	RBF	↓**70.0** ± 1.3	↓**69.5** ± 2.8	70.6 ± 2.7	70.3 ± 1.7	↓**0.699** ± 0.015
Transformed link-based features	Bagging	*↑**88.3** ± 1.1	91.3 ± 1.6	85.3 ± 1.2	↑**86.2** ± 1.1	*↑**0.886** ± 0.011
	MLP-GD	87.4 ± 1.6	89.6 ± 2.2	85.2 ± 1.1	85.9 ± 2.0	0.877 ± 0.019
	Random forest	87.4 ± 1.4	89.5 ± 3.7	85.3 ± 2.4	86.0 ± 1.7	0.876 ± 0.016
	AdaBoost	87.1 ± 1.2	88.8 ± 0.9	↑**85.5** ± 2.0	85.9 ± 1.7	0.873 ± 0.011
	MLP-LM	86.3 ± 2.7	87.5 ± 4.0	85.2 ± 3.6	85.8 ± 3.3	0.866 ± 0.027
	SVM	86.0 ± 0.7	88.5 ± 1.7	83.5 ± 1.8	84.4 ± 1.3	0.864 ± 0.007
	LogitBoost	84.4 ± 2.2	87.6 ± 3.7	81.3 ± 1.3	82.7 ± 1.0	0.850 ± 0.021
	C4.5	84.1 ± 1.9	85.6 ± 2.4	82.5 ± 1.8	83.1 ± 1.8	0.844 ± 0.020
	SOM + LVQ	85.2 ± 2.1	↓**81.3** ± 2.1	83.3 ± 1.1	82.1 ± 1.5	0.836 ± 0.011
	OneR	82.7 ± 0.7	86.5 ± 2.0	79.0 ± 1.9	80.5 ± 1.2	0.834 ± 0.008
	RBF	81.3 ± 0.9	82.0 ± 2.2	80.6 ± 2.0	80.9 ± 1.4	0.814 ± 0.010
	IBK	80.7 ± 0.1	81.6 ± 0.7	79.8 ± 0.6	80.2 ± 0.4	0.809 ± 0.002
	Naive Bayes	*↓**61.0** ± 4.7	*↑**96.8** ± 0.8	*↓**25.2** ± 9.6	*↓**56.6** ± 3.2	↓**0.714** ± 0.025

features is more effective than use individual feature set. The combination that acquired the best results was content and transformed link-based features with balanced classes. However, in the experiments with unbalanced classes, the combination of content and link-based features was more effective. The results also indicate that the combination of all feature sets also achieved good results. Nevertheless, such combination has the largest dimensionality and thus requires more computational resources.

Again, we observed that the learning methods achieved better performance with balanced classes. For example, in the experiments shown in Table 6, for the combination of content and link-based features, the precision and recall rate were on average, respectively, 9.6% and 14.8% higher than the results achieved in the same experiment with unbalanced classes.

TABLE 5: Results achieved by each classifier using combination of features extracted from WEBSPAM-UK2006 dataset with unbalanced classes.

		Accuracy	Recall	Specificity	Precision	F-measure
Combination of content and link-based features	AdaBoost	*↑**92.5** ± 0.6	80.4 ± 1.9	↑**96.1** ± 0.4	*↑**86.3** ± 1.2	*↑**0.832** ± 0.014
	MLP-LM	92.1 ± 0.6	81.7 ± 2.0	95.3 ± 0.2	84.4 ± 1.9	0.830 ± 0.008
	Bagging	92.0 ± 0.7	81.1 ± 1.8	95.3 ± 0.6	84.0 ± 1.7	0.825 ± 0.016
	Random forest	91.0 ± 0.3	74.4 ± 2.5	96.0 ± 1.1	85.0 ± 2.4	0.793 ± 0.006
	MLP-GD	89.3 ± 0.6	74.0 ± 3.4	94.1 ± 0.9	79.7 ± 2.7	0.767 ± 0.017
	C4.5	88.2 ± 0.5	75.9 ± 2.3	91.9 ± 0.3	73.8 ± 0.9	0.748 ± 0.014
	LogitBoost	88.5 ± 0.6	70.2 ± 2.6	93.9 ± 0.7	77.7 ± 2.3	0.737 ± 0.019
	IBK	86.7 ± 0.6	68.7 ± 2.3	92.1 ± 0.7	72.5 ± 2.0	0.705 ± 0.015
	Naive Bayes	72.7 ± 0.9	95.3 ± 1.0	↓**65.8** ± 1.3	↓**45.9** ± 0.8	0.620 ± 0.008
	SVM	83.6 ± 0.8	52.4 ± 2.0	93.0 ± 0.8	69.6 ± 2.5	0.598 ± 0.018
	OneR	78.5 ± 0.9	50.1 ± 2.4	87.2 ± 1.2	54.4 ± 2.2	0.521 ± 0.018
	SOM + LVQ	*↓**24.2** ± 2.3	↑**95.7** ± 0.9	79.1 ± 0.7	63.5 ± 4.7	0.349 ± 0.027
	RBF	77.8 ± 0.5	*↓**20.1** ± 1.0	95.2 ± 0.4	56.3 ± 2.7	*↓**0.296** ± 0.014
Combination of link and transformed link-based features	Bagging	↑**89.7** ± 0.8	78.5 ± 3.0	93.1 ± 0.5	↑**77.6** ± 1.5	↑**0.780** ± 0.020
	Random forest	89.0 ± 1.0	78.7 ± 2.2	92.2 ± 0.9	75.4 ± 2.3	0.770 ± 0.020
	MLP-GD	89.1 ± 1.5	76.9 ± 3.8	92.7 ± 1.5	76.1 ± 4.5	0.764 ± 0.031
	MLP-LM	88.8 ± 1.8	75.8 ± 4.7	92.9 ± 1.2	76.6 ± 4.1	0.761 ± 0.038
	AdaBoost	88.3 ± 0.6	73.9 ± 2.2	92.6 ± 0.5	75.3 ± 1.3	0.746 ± 0.015
	C4.5	86.4 ± 0.9	71.8 ± 3.3	90.9 ± 1.1	70.6 ± 2.2	0.712 ± 0.019
	LogitBoost	86.4 ± 0.7	69.9 ± 4.5	91.4 ± 1.7	71.4 ± 2.7	0.705 ± 0.018
	IBK	84.3 ± 0.8	67.2 ± 2.0	89.5 ± 0.7	66.0 ± 1.8	0.666 ± 0.017
	SVM	83.6 ± 1.0	62.5 ± 2.1	90.0 ± 1.3	65.6 ± 2.8	0.639 ± 0.018
	Naive Bayes	73.5 ± 0.8	↑**94.9** ± 1.1	↓**67.0** ± 1.0	↓**46.7** ± 0.8	0.626 ± 0.008
	OneR	82.4 ± 0.7	61.6 ± 2.0	88.7 ± 0.7	62.4 ± 1.6	0.620 ± 0.015
	SOM + LVQ	↓**27.6** ± 2.6	94.3 ± 0.7	78.8 ± 0.5	59.6 ± 2.3	0.377 ± 0.025
	RBF	78.4 ± 0.8	↓**22.6** ± 1.2	↑**95.4** ± 0.8	60.0 ± 4.9	↓**0.328** ± 0.019
Combination of content and transformed link-based features	AdaBoost	↑**92.4** ± 0.8	81.1 ± 2.7	95.8 ± 0.5	↑**85.6** ± 1.6	*↑**0.832** ± 0.018
	Bagging	92.1 ± 0.7	82.2 ± 2.4	95.1 ± 0.4	83.6 ± 1.4	0.829 ± 0.017
	Random forest	91.0 ± 0.3	81.0 ± 2.0	94.1 ± 0.3	80.7 ± 0.6	0.808 ± 0.010
	MLP-GD	90.3 ± 0.8	78.5 ± 2.1	94.0 ± 0.8	79.8 ± 3.2	0.791 ± 0.021
	MLP-LM	89.9 ± 1.1	77.3 ± 4.3	93.7 ± 1.5	79.0 ± 4.2	0.780 ± 0.032
	LogitBoost	88.9 ± 0.5	73.0 ± 3.5	93.7 ± 0.8	77.9 ± 1.7	0.753 ± 0.016
	C4.5	87.5 ± 0.8	72.9 ± 1.8	91.9 ± 1.0	73.4 ± 2.3	0.731 ± 0.015
	IBK	87.1 ± 0.4	68.0 ± 1.8	92.9 ± 0.6	74.5 ± 1.4	0.711 ± 0.010
	OneR	83.4 ± 1.3	63.3 ± 2.1	89.5 ± 1.3	64.9 ± 3.3	0.641 ± 0.024
	SVM	85.1 ± 0.6	50.7 ± 2.0	95.6 ± 0.5	77.7 ± 2.2	0.613 ± 0.018
	SOM + LVQ	↓**33.2** ± 3.8	*↑**96.0** ± 0.6	81.4 ± 0.7	71.5 ± 2.5	0.452 ± 0.037
	RBF	81.0 ± 0.6	↓**29.9** ± 2.4	*↑**96.5** ± 0.4	71.9 ± 2.8	0.422 ± 0.027
	Naive Bayes	36.1 ± 0.9	*↑**96.0** ± 1.1	*↓**17.9** ± 1.1	*↓**26.2** ± 0.3	↓**0.412** ± 0.005
Combination of all features	AdaBoost	↑**92.1** ± 0.7	81.6 ± 1.2	95.3 ± 0.9	↑**84.2** ± 2.7	↑**0.829** ± 0.014
	Bagging	91.8 ± 1.0	82.1 ± 2.0	94.8 ± 1.0	82.7 ± 2.9	0.824 ± 0.019
	Random forest	91.2 ± 0.7	82.4 ± 1.1	93.9 ± 0.9	80.4 ± 2.3	0.814 ± 0.012
	MLP-LM	90.6 ± 1.0	81.7 ± 3.7	93.3 ± 0.7	79.2 ± 2.5	0.804 ± 0.027
	MLP-GD	90.2 ± 1.0	79.9 ± 2.7	93.4 ± 0.9	78.9 ± 3.3	0.793 ± 0.024
	LogitBoost	89.3 ± 0.9	75.9 ± 2.3	93.3 ± 0.8	77.6 ± 2.2	0.767 ± 0.020
	C4.5	87.4 ± 0.6	73.4 ± 2.7	91.7 ± 1.0	73.0 ± 2.0	0.731 ± 0.014
	IBK	86.4 ± 0.6	69.2 ± 1.4	91.7 ± 0.6	71.6 ± 1.6	0.704 ± 0.013
	OneR	82.9 ± 0.7	63.5 ± 2.5	88.8 ± 0.8	63.3 ± 1.6	0.634 ± 0.017
	SVM	84.4 ± 0.7	55.9 ± 2.0	93.1 ± 0.5	71.1 ± 1.9	0.626 ± 0.018
	Naive Bayes	73.4 ± 0.7	94.8 ± 1.2	↓**66.9** ± 1.0	↓**46.6** ± 0.7	0.625 ± 0.006
	SOM + LVQ	↓**25.5** ± 4.1	↑**95.4** ± 1.6	79.2 ± 0.8	63.8 ± 6.0	0.361 ± 0.040
	RBF	78.3 ± 0.5	↓**20.2** ± 3.2	↑**95.9** ± 0.4	60.0 ± 2.5	↓**0.302** ± 0.039

TABLE 6: Results achieved by each classifier using combination of features extracted from WEBSPAM-UK2006 dataset with balanced classes.

		Accuracy	Recall	Specificity	Precision	F-measure
Combination of content and link-based features	AdaBoost	↑**90.3** ± 0.6	92.5 ± 1.3	↑**88.1** ± 1.3	↑**88.7** ± 1.0	↑**0.905** ± 0.006
	Bagging	90.1 ± 0.6	↑**93.2** ± 0.8	87.1 ± 1.2	87.8 ± 1.0	0.904 ± 0.005
	MLP-GD	89.2 ± 1.0	93.0 ± 1.7	85.5 ± 2.2	86.4 ± 2.2	0.895 ± 0.011
	Random forest	89.0 ± 0.7	92.1 ± 2.1	85.7 ± 1.9	86.9 ± 1.2	0.894 ± 0.008
	MLP-LM	88.4 ± 2.1	91.9 ± 3.5	85.0 ± 3.3	85.6 ± 2.7	0.886 ± 0.021
	LogitBoost	86.4 ± 0.5	89.9 ± 1.3	82.9 ± 2.0	84.3 ± 1.2	0.870 ± 0.005
	C4.5	85.7 ± 0.8	86.9 ± 2.2	84.5 ± 1.4	85.1 ± 0.9	0.860 ± 0.009
	IBK	83.1 ± 0.8	84.7 ± 2.7	81.5 ± 1.6	82.4 ± 1.0	0.835 ± 0.012
	Naive Bayes	80.8 ± 1.2	92.7 ± 6.5	68.9 ± 6.7	75.2 ± 3.5	0.828 ± 0.016
	OneR	81.5 ± 1.2	88.8 ± 1.7	74.2 ± 1.8	77.5 ± 1.3	0.827 ± 0.012
	SVM	76.0 ± 1.5	71.4 ± 1.7	80.6 ± 2.6	78.7 ± 2.3	0.749 ± 0.015
	RBF	68.7 ± 2.3	↓**65.0** ± 1.6	72.4 ± 4.5	70.3 ± 3.4	0.675 ± 0.018
	SOM + LVQ	↓**64.6** ± 3.6	70.7 ± 3.8	↓**67.7** ± 2.4	↓**68.9** ± 3.0	↓**0.666** ± 0.026
Combination of link and transformed link-based features	Bagging	↑**88.2** ± 1.2	91.6 ± 1.3	84.9 ± 1.9	↑**85.8** ± 1.6	↑**0.886** ± 0.011
	Random forest	87.7 ± 0.6	92.1 ± 0.6	83.3 ± 1.5	84.7 ± 1.1	0.882 ± 0.005
	MLP-GD	87.6 ± 2.5	91.7 ± 1.6	83.4 ± 3.9	84.6 ± 3.4	0.880 ± 0.023
	AdaBoost	86.9 ± 0.9	89.7 ± 0.9	84.2 ± 1.4	85.0 ± 1.2	0.873 ± 0.008
	LogitBoost	85.8 ± 0.8	90.3 ± 1.3	81.4 ± 2.2	83.0 ± 1.5	0.864 ± 0.006
	MLP-LM	86.7 ± 2.7	87.7 ± 2.9	↑**85.8** ± 3.7	85.2 ± 3.5	0.863 ± 0.026
	OneR	84.0 ± 0.9	87.8 ± 2.0	80.1 ± 1.9	81.6 ± 1.3	0.846 ± 0.009
	C4.5	84.2 ± 1.1	84.7 ± 1.5	83.6 ± 1.5	83.8 ± 1.3	0.842 ± 0.011
	Naive Bayes	80.8 ± 1.5	↑**94.7** ± 0.7	↓**66.9** ± 3.0	74.1 ± 1.7	0.832 ± 0.011
	SVM	80.9 ± 1.0	84.0 ± 1.7	77.7 ± 1.2	79.0 ± 0.9	0.814 ± 0.010
	IBK	80.6 ± 1.4	83.2 ± 1.6	78.0 ± 1.6	79.1 ± 1.4	0.811 ± 0.014
	SOM + LVQ	↓**69.8** ± 3.7	72.7 ± 1.9	71.3 ± 1.7	71.9 ± 1.4	0.708 ± 0.023
	RBF	69.9 ± 2.0	↓**68.5** ± 1.7	71.3 ± 3.4	↓**70.6** ± 2.6	↓**0.695** ± 0.018
Combination of content and transformed link-based features	AdaBoost	*↑**91.4** ± 0.7	93.7 ± 1.2	↑**89.2** ± 1.4	↑**89.7** ± 1.1	*↑**0.916** ± 0.007
	Bagging	90.0 ± 0.8	92.7 ± 1.1	87.3 ± 1.2	88.0 ± 1.0	0.903 ± 0.008
	MLP-GD	89.6 ± 1.6	91.2 ± 2.0	88.0 ± 2.4	88.2 ± 2.6	0.897 ± 0.016
	Random forest	89.1 ± 0.7	92.5 ± 0.8	85.6 ± 1.1	86.6 ± 0.9	0.894 ± 0.006
	MLP-LM	89.1 ± 1.8	90.5 ± 2.9	87.8 ± 2.8	88.2 ± 2.1	0.893 ± 0.017
	LogitBoost	87.2 ± 1.1	90.8 ± 1.7	83.7 ± 2.1	84.8 ± 1.6	0.877 ± 0.011
	C4.5	85.9 ± 1.1	86.8 ± 1.5	85.0 ± 1.9	85.3 ± 1.5	0.860 ± 0.010
	OneR	83.1 ± 0.7	87.3 ± 2.0	78.9 ± 1.8	80.6 ± 1.1	0.838 ± 0.008
	IBK	83.3 ± 1.1	81.5 ± 2.0	85.1 ± 1.5	84.6 ± 1.3	0.830 ± 0.012
	SVM	77.9 ± 1.1	66.5 ± 2.4	↑**89.2** ± 1.7	86.1 ± 1.7	0.750 ± 0.015
	Naive Bayes	*↓**61.7** ± 4.7	*↑**96.5** ± 0.9	*↓**26.9** ± 9.9	*↓**57.1** ± 3.5	0.717 ± 0.026
	SOM + LVQ	63.8 ± 4.6	77.4 ± 2.9	70.6 ± 1.7	73.9 ± 1.9	0.684 ± 0.026
	RBF	69.1 ± 1.2	*↓**51.6** ± 3.1	86.6 ± 3.1	79.6 ± 3.2	*↓**0.625** ± 0.019
Combination of all features	AdaBoost	↑**91.3** ± 1.0	93.2 ± 0.9	*↑**89.4** ± 1.5	*↑**89.8** ± 1.3	↑**0.915** ± 0.010
	MLP-GD	90.4 ± 1.0	92.4 ± 1.4	88.3 ± 1.6	88.8 ± 1.5	0.906 ± 0.010
	Bagging	90.2 ± 1.1	92.5 ± 1.3	87.9 ± 1.6	88.4 ± 1.3	0.904 ± 0.010
	Random forest	89.4 ± 0.8	92.7 ± 1.5	86.1 ± 1.6	87.0 ± 1.2	0.897 ± 0.008
	MLP-LM	87.5 ± 3.7	89.7 ± 4.6	85.2 ± 5.4	86.1 ± 4.0	0.878 ± 0.034
	LogitBoost	86.9 ± 1.1	89.8 ± 2.2	84.0 ± 1.9	84.9 ± 1.5	0.873 ± 0.011
	C4.5	85.6 ± 1.6	86.3 ± 2.6	84.9 ± 2.0	85.1 ± 1.7	0.857 ± 0.017
	OneR	83.1 ± 0.8	86.8 ± 2.0	79.4 ± 1.4	80.9 ± 0.9	0.837 ± 0.009
	Naive Bayes	81.0 ± 1.5	↑**93.8** ± 1.2	↓**68.2** ± 2.6	74.7 ± 1.6	0.832 ± 0.012
	IBK	83.0 ± 1.7	83.0 ± 2.3	82.9 ± 2.5	83.0 ± 2.1	0.830 ± 0.017
	SVM	80.4 ± 1.1	78.2 ± 1.4	82.6 ± 1.9	81.9 ± 1.5	0.800 ± 0.011
	SOM + LVQ	↓**67.1** ± 3.2	71.0 ± 3.7	69.0 ± 1.3	↓**69.9** ± 2.1	0.684 ± 0.015
	RBF	69.5 ± 1.7	↓**64.6** ± 1.5	74.5 ± 2.7	71.8 ± 2.3	↓**0.679** ± 0.016

The classifier that achieved the best overall results was the AdaBoost. The bagging method performed better than AdaBoost just for the combination of link and transformed link-based features (Table 5). The MLP neural networks and random forest also achieved good results, staying in the top five methods. On the other hand, the RBF neural network and SOM + LVQ achieved the worst results and were inferior than one of the simplest pattern classification methods, the OneR.

5.2 Results under WEBSPAM-UK2007

This section presents the results achieved by the machine learning methods described in Section 3 using sets of pre-computed features vectors extracted from WEBSPAM-UK2007 dataset and made available by the organizers of the Web Spam Challenge 2008. Each set contains 5,476 (94.5%) samples of ham hosts and 321 (5.5%) of spam ones.

We can note that the imbalance class problem for the WEBSPAM-UK2007 dataset is much worse than WEBSPAM-UK2006. It was infeasible to use only the random undersampling method to balance the classes because would be necessary to discard 5,155 samples of ham. This high number of discarded instances could affect the learning process of the classifiers due to the great loss of information. Therefore, we used the synthetic minority oversampling technique (SMOTE) [39]. This is a well-known balancing method that creates synthetic instances of the minority class using knowledge of the existing instances.

Using only the SMOTE method would be necessary to create 5,155 synthetic instances of spam. This high number could cause overfitting. In order to avoid this, we used the SMOTE to create 1,605 new samples of the spam and we applied the random undersampling method in each simulation discarding 3,550 instances of the ham class. In this way, we kept the classes balanced with 1,926 representatives each one.

After applying the SMOTE and random undersampling methods was possible to perform experiments using balanced classes. However, it was infeasible to perform experiments with unbalanced classes because some classification methods were not able to detect any spam instance during the test step, since only 321 instances were insufficient for learning. For this reason, for the WEBSPAM-UK2007 dataset we performed experiments only with balanced classes.

Table 7 presents the results achieved by each classifier exploring the content, links and transformed links-based features, respectively. The results are sorted by F-measure for each feature set. Each column of the table shows the average and the standard deviation of the results achieved using the random sub-sampling validation with 10 iterations. The bold values preceded by the symbol "↑" indicate the highest score and the bold values preceded by the symbol "↓" indicate the lowest score for each performance measure. Further, values preceded by the symbol "*" indicate the highest or lowest score considering the three set of features.

The results indicate that the evaluated methods acquired better performance in the simulations with content-based features. Moreover, the results presented in Table 7 indicate that the bagging of decision trees has outperformed other methods. We can note that in all scenarios it achieved the best accuracy, precision and f-measure rates.

5.2.1 Combinations of feature vectors extracted from WEBSPAM-UK2007 dataset.

This section presents the results achieved by the combinations of features. As previously described,

TABLE 7: Results achieved by each classifier using features extracted from WEBSPAM-UK2007 dataset with balanced classes.

		Accuracy	Recall	Specificity	Precision	F-measure
Content-based features	Bagging	***↑89,7** ± 1,3	89,2 ± 1,3	90,2 ± 1,7	***↑90,0** ± 2,0	***↑0,896** ± 0,014
	AdaBoost	88,0 ± 2,2	88,1 ± 2,9	87,9 ± 2,3	87,9 ± 2,2	0,880 ± 0,022
	Random forest	86,6 ± 1,2	89,5 ± 1,4	83,8 ± 1,7	84,6 ± 1,4	0,870 ± 0,011
	IBK	85,8 ± 1,3	83,8 ± 2,7	87,8 ± 2,0	87,3 ± 1,6	0,855 ± 0,015
	MLP-LM	82,9 ± 2,1	83,1 ± 4,1	82,5 ± 4,0	82,2 ± 2,5	0,826 ± 0,023
	C4.5	79,6 ± 1,6	79,6 ± 1,8	79,6 ± 2,5	79,6 ± 2,1	0,796 ± 0,015
	OneR	80,1 ± 2,2	74,8 ± 4,6	85,3 ± 4,1	83,7 ± 3,6	0,789 ± 0,026
	MLP-GD	76,2 ± 1,8	74,8 ± 2,9	77,7 ± 4,3	77,2 ± 3,3	0,759 ± 0,019
	OneR	75,0 ± 0,9	75,3 ± 2,9	74,6 ± 3,0	74,8 ± 1,7	0,750 ± 0,011
	Naive Bayes	***↓52,6** ± 2,0	***↑95,0** ± 2,3	***↓10,2** ± 4,4	***↓51,4** ± 1,1	0,667 ± 0,011
	SOM + LVQ	56,7 ± 2,3	83,0 ± 3,2	69,9 ± 1,4	77,1 ± 3,2	0,653 ± 0,016
	RBF	68,8 ± 2,3	53,4 ± 3,0	84,3 ± 2,3	77,3 ± 3,1	0,631 ± 0,029
	SVM	70,6 ± 0,9	**↓49,8** ± 2,0	***↑91,5** ± 1,1	85,4 ± 1,5	**↓0,628** ± 0,016
Link-based features	Bagging	**↑88,5** ± 1,6	87,6 ± 1,5	**↑89,4** ± 2,2	**↑89,2** ± 2,1	**↑0,884** ± 0,015
	Random forest	84,7 ± 1,3	**↑88,9** ± 1,4	80,5 ± 2,3	82,0 ± 1,7	0,853 ± 0,011
	AdaBoost	85,2 ± 2,0	85,4 ± 2,0	84,9 ± 3,3	85,0 ± 2,9	0,852 ± 0,019
	OneR	80,9 ± 2,7	80,4 ± 7,0	81,3 ± 8,9	82,1 ± 7,7	0,807 ± 0,025
	MLP-LM	80,4 ± 1,4	78,6 ± 3,7	82,3 ± 3,7	82,2 ± 3,3	0,803 ± 0,012
	IBK	79,9 ± 0,9	78,2 ± 2,2	81,6 ± 1,8	81,0 ± 1,3	0,795 ± 0,011
	SVM	80,0 ± 1,1	75,7 ± 4,2	84,4 ± 4,2	83,1 ± 3,3	0,791 ± 0,015
	C4.5	78,8 ± 1,4	78,5 ± 2,6	79,1 ± 2,6	79,0 ± 1,9	0,787 ± 0,015
	OneR	70,8 ± 1,6	71,5 ± 4,0	70,0 ± 3,8	70,5 ± 2,1	0,709 ± 0,020
	MLP-GD	66,7 ± 2,2	66,3 ± 3,5	**↓67,2** ± 3,2	**↓67,5** ± 3,1	0,668 ± 0,024
	RBF	70,1 ± 1,7	52,3 ± 3,3	88,0 ± 2,3	81,4 ± 2,8	0,636 ± 0,027
	SOM + LVQ	**↓53,3** ± 2,2	82,1 ± 3,4	67,7 ± 1,6	75,0 ± 3,4	0,622 ± 0,017
	Naive Bayes	54,1 ± 3,4	***↓20,3** ± 24,3	87,8 ± 17,9	67,8 ± 9,3	***↓0,250** ± 0,201
Transformed link-based features	Bagging	**↑87,0** ± 1,2	89,2 ± 1,6	84,7 ± 2,3	**↑85,4** ± 1,8	**↑0,872** ± 0,012
	Random forest	84,5 ± 1,2	**↑90,6** ± 1,5	78,5 ± 2,4	80,8 ± 1,7	0,854 ± 0,010
	IBK	84,9 ± 1,1	88,2 ± 1,2	81,6 ± 1,9	82,7 ± 1,5	0,853 ± 0,010
	SVM	85,1 ± 1,2	85,2 ± 2,1	85,0 ± 1,8	85,1 ± 1,5	0,851 ± 0,012
	AdaBoost	83,8 ± 1,5	85,0 ± 2,2	82,6 ± 2,2	83,0 ± 1,9	0,840 ± 0,015
	OneR	81,3 ± 1,7	84,4 ± 2,9	78,2 ± 2,3	79,4 ± 1,8	0,818 ± 0,018
	MLP-LM	81,4 ± 3,3	81,2 ± 5,9	81,9 ± 2,7	81,8 ± 3,6	0,814 ± 0,038
	C4.5	78,5 ± 1,5	79,9 ± 1,5	77,2 ± 3,0	77,8 ± 2,2	0,788 ± 0,013
	MLP-GD	74,1 ± 1,3	74,8 ± 1,6	73,4 ± 2,8	74,4 ± 2,6	0,746 ± 0,015
	OneR	71,6 ± 2,0	68,9 ± 3,3	74,4 ± 3,4	72,9 ± 2,6	0,708 ± 0,022
	SOM + LVQ	71,2 ± 2,9	69,3 ± 2,8	70,3 ± 1,7	69,9 ± 1,9	0,705 ± 0,018
	RBF	64,6 ± 1,8	59,6 ± 4,2	**↓69,6** ± 2,4	**↓66,2** ± 1,7	0,627 ± 0,027
	Naive Bayes	**↓58,6** ± 1,8	**↓28,2** ± 5,9	**↑89,0** ± 3,2	72,0 ± 2,8	**↓0,402** ± 0,058

we have performed the experiments with balanced classes.

Table 8 presents the results sorted by F-measure for each used feature set. The bold values preceded by the symbol "↑" indicate the highest score and the bold values preceded by the symbol "↓" indicate the lowest score for each performance measure. Further, values preceded by the symbol "*" indicate the highest or lowest score considering all features combinations.

The results indicate that the combination of feature vectors improves the classifiers performance. Only the combination of links and transformed link-based features did not achieved better results than the ones achieved without combination of features. Furthermore, as well as in the experiments with WEBSPAM-UK2006 dataset, the classifiers performed better using the combination of content

TABLE 8: Results achieved by each classifier using combination of features extracted from WEBSPAM-UK2007 dataset with balanced classes.

		Accuracy	Recall	Specificity	Precision	F-measure
Combination of content and link-based features	AdaBoost	↑**93,5** ± 0,5	↑**94,5** ± 0,7	↑**92,6** ± 1,2	↑**92,7** ± 1,1	↑**0,936** ± 0,005
	Bagging	89,5 ± 1,3	91,4 ± 1,1	87,7 ± 2,4	88,1 ± 2,0	0,897 ± 0,012
	MLP-LM	87,2 ± 2,4	87,2 ± 3,2	87,3 ± 3,3	86,3 ± 4,4	0,866 ± 0,029
	Random forest	85,8 ± 1,0	90,3 ± 1,8	81,3 ± 1,5	82,8 ± 1,1	0,864 ± 0,010
	IBK	85,6 ± 1,5	85,9 ± 2,1	85,4 ± 1,4	85,4 ± 1,4	0,857 ± 0,016
	OneR	84,3 ± 1,9	79,0 ± 4,0	89,5 ± 5,7	88,7 ± 5,5	0,834 ± 0,017
	C4.5	79,4 ± 1,7	79,0 ± 1,6	79,8 ± 3,2	79,7 ± 2,5	0,793 ± 0,015
	MLP-GD	78,4 ± 1,8	78,5 ± 3,5	78,4 ± 2,5	78,1 ± 2,6	0,783 ± 0,020
	SVM	78,2 ± 1,6	72,6 ± 2,6	83,6 ± 1,6	81,6 ± 1,6	0,769 ± 0,019
	OneR	76,3 ± 1,1	75,5 ± 3,4	77,1 ± 3,3	76,8 ± 2,0	0,761 ± 0,014
	Naive Bayes	↓**59,5** ± 4,8	81,6 ± 19,0	*↓**37,4** ± 25,7	*↓**58,8** ± 8,7	0,660 ± 0,057
	SOM + LVQ	61,2 ± 3,6	73,9 ± 4,3	67,6 ± 1,5	70,3 ± 2,7	0,653 ± 0,018
	RBF	65,8 ± 1,3	↓**58,8** ± 1,6	72,9 ± 3,0	68,5 ± 2,1	↓**0,632** ± 0,011
Combination of link and transformed link-based features	Bagging	↑**88,7** ± 1,5	86,5 ± 2,2	↑**90,9** ± 2,0	↑**90,5** ± 1,9	↑**0,885** ± 0,016
	Random forest	86,4 ± 1,0	↑**89,5** ± 1,3	83,3 ± 2,1	84,3 ± 1,6	0,868 ± 0,009
	IBK	86,1 ± 0,8	87,2 ± 0,6	85,1 ± 1,5	85,4 ± 1,2	0,863 ± 0,007
	AdaBoost	85,5 ± 0,7	84,9 ± 1,5	86,2 ± 1,0	86,0 ± 0,8	0,854 ± 0,008
	MLP-LM	83,4 ± 2,8	83,2 ± 4,4	83,8 ± 4,0	84,2 ± 3,4	0,836 ± 0,028
	C4.5	81,3 ± 1,2	80,3 ± 2,6	82,3 ± 1,3	81,9 ± 1,0	0,811 ± 0,014
	OneR	81,1 ± 0,8	78,5 ± 6,2	83,6 ± 6,5	83,3 ± 5,2	0,805 ± 0,014
	MLP-GD	79,8 ± 2,2	78,5 ± 3,1	81,2 ± 2,2	80,9 ± 2,5	0,797 ± 0,025
	SVM	79,8 ± 1,5	73,0 ± 4,5	86,6 ± 2,8	84,6 ± 2,1	0,783 ± 0,021
	OneR	76,7 ± 1,7	78,7 ± 3,6	74,6 ± 2,9	75,6 ± 1,9	0,771 ± 0,020
	SOM + LVQ	62,7 ± 3,2	76,2 ± 2,5	↓**69,5** ± 1,8	72,5 ± 2,2	0,672 ± 0,022
	RBF	67,8 ± 2,7	57,6 ± 1,9	78,0 ± 4,1	72,5 ± 4,1	0,641 ± 0,025
	Naive Bayes	*↓**56,3** ± 1,5	*↓**23,8** ± 3,2	88,6 ± 2,4	↓**67,8** ± 4,5	*↓**0,352** ± 0,036
Combination of content and transformed link-based features	AdaBoost	*↑**94,4** ± 0,7	*↑**94,9** ± 0,7	↑**94,0** ± 1,3	↑**94,0** ± 1,2	*↑**0,944** ± 0,007
	Bagging	92,2 ± 0,9	92,8 ± 1,4	91,7 ± 0,7	91,8 ± 0,7	0,923 ± 0,009
	Random forest	88,0 ± 1,1	91,9 ± 0,9	84,0 ± 1,7	85,1 ± 1,4	0,884 ± 0,010
	IBK	87,9 ± 1,5	88,9 ± 2,0	86,9 ± 1,9	87,1 ± 1,6	0,880 ± 0,015
	MLP-LM	85,7 ± 2,4	85,3 ± 3,0	86,1 ± 2,8	85,8 ± 2,8	0,855 ± 0,025
	MLP-GD	82,7 ± 2,8	84,0 ± 3,2	81,4 ± 3,5	81,6 ± 4,3	0,827 ± 0,033
	C4.5	82,3 ± 1,4	82,4 ± 2,8	82,2 ± 2,0	82,2 ± 1,5	0,823 ± 0,015
	OneR	82,9 ± 2,4	76,1 ± 2,7	89,7 ± 6,1	88,6 ± 6,1	0,817 ± 0,020
	OneR	79,1 ± 1,6	79,9 ± 2,8	78,3 ± 1,4	78,6 ± 1,3	0,792 ± 0,018
	SVM	72,5 ± 1,1	60,7 ± 3,0	84,3 ± 2,3	79,5 ± 2,0	0,688 ± 0,018
	Naive Bayes	69,3 ± 3,8	63,8 ± 12,1	74,8 ± 10,1	↓**72,4** ± 4,6	0,670 ± 0,066
	SOM + LVQ	↓**58,0** ± 6,2	81,0 ± 3,8	↓**69,5** ± 1,9	75,5 ± 2,5	0,654 ± 0,036
	RBF	67,9 ± 1,8	↓**53,3** ± 2,6	82,6 ± 1,8	75,4 ± 2,4	↓**0,624** ± 0,024
Combination of all features	AdaBoost	↑**94,1** ± 0,8	↑**94,0** ± 1,0	*↑**94,2** ± 1,2	*↑**94,2** ± 1,2	↑**0,941** ± 0,008
	Bagging	90,9 ± 1,2	91,0 ± 1,7	90,8 ± 1,6	90,8 ± 1,5	0,909 ± 0,012
	IBK	88,9 ± 1,2	90,6 ± 1,9	87,2 ± 2,1	87,6 ± 1,7	0,890 ± 0,012
	Random forest	87,8 ± 1,0	91,2 ± 1,4	84,4 ± 1,7	85,3 ± 1,3	0,881 ± 0,010
	MLP-LM	87,9 ± 1,4	88,4 ± 2,8	87,3 ± 2,7	87,1 ± 2,4	0,877 ± 0,016
	MLP-GD	83,7 ± 2,0	85,5 ± 2,3	81,9 ± 2,8	83,1 ± 3,1	0,842 ± 0,020
	OneR	84,1 ± 1,9	77,0 ± 3,6	91,1 ± 4,6	90,0 ± 4,7	0,828 ± 0,019
	C4.5	82,5 ± 1,3	82,6 ± 3,3	82,5 ± 1,4	82,5 ± 0,9	0,825 ± 0,016
	OneR	79,3 ± 1,4	79,2 ± 2,9	79,4 ± 1,4	79,3 ± 1,1	0,792 ± 0,017
	SVM	78,6 ± 1,4	72,1 ± 2,4	85,1 ± 2,2	82,9 ± 2,0	0,771 ± 0,017
	Naive Bayes	66,6 ± 5,6	69,8 ± 11,1	↓**63,4** ± 20,3	↓**67,8** ± 8,9	0,675 ± 0,032
	RBF	68,0 ± 1,6	↓**60,7** ± 1,8	75,4 ± 2,1	71,2 ± 2,1	0,655 ± 0,017
	SOM + LVQ	↓**58,4** ± 3,0	76,1 ± 3,5	67,2 ± 1,5	71,0 ± 2,7	↓**0,640** ± 0,018

and transformed link-based features. However, the results achieved with the combination of all features were also satisfactory but the high dimensional feature space (275 dimensions) requires more computational cost.

Regarding the methods, the AdaBoost achieved the best results and the bagging was the second best method with no significant statistical difference. On the other hand, the group of the four methods with the worst performance is composed by the naive Bayes, RBF, SOM + LVQ and LogitBoost.

5.3 Statistical analysis of results

In order to support our claims, we also performed a statistical analysis of the results (Table 9). For that, we ranked the methods by F-measure using the Wilcoxon rank-sum test [40] and 95% of confidence interval. Such method is a statistical hypothesis test which is also sometimes called the Mann-Whitney test [40]. The methods that are at the same level in the table have statistically equivalent results.

Note that, for WEBSPAM-UK2006 dataset with unbalanced classes, the bagging of decision trees is statistically equal to random forest and superior than other evaluated methods. However, in the scenario with balanced classes it is statistically equal to AdaBoost, MLP neural networks and random forest. On the other hand, the OneR, SVM, naive Bayes, SOM + LVQ and RBF are statistically inferior than all evaluated methods.

In the experiments with WEBSPAM-UK2007 dataset, the statistical analysis shows that the bagging and AdaBoost were superior than other evaluated learning algorithms. Nevertheless, the LogitBoost, SOM + LVQ, RBF neural network and naive Bayes were statistically inferior.

TABLE 9: Statistical analysis of the results using the Wilcoxon rank-sum test

Results achieved for WEBSPAM-UK2006 dataset		
Level	Methods	
	Unbalanced classes	**Balanced classes**
1	Bagging and Random forest	Bagging, AdaBoost, Random forest, MLP-GD and MLP-LM
2	Random forest, AdaBoost and MLP-LM	C4.5 and LogitBoost
3	MLP-GD	IBK
4	C4.5 and LogitBoost	OneR
5	IBK	SVM and Naive Bayes
6	SVM and RBF	SOM+LVQ
7	OneR	RBF
8	Naive Bayes	
9	SOM + LVQ	
10		

Results achieved for WEBSPAM-UK2007 dataset	
Level	Methods
	Balanced classes
1	Bagging and AdaBoost
2	Random forest and IBK
3	MLP-LM
4	OneR
5	C4.5
6	MLP-GD and SVM
7	LogitBoost
8	SOM + LVQ
9	RBF
10	Naive Bayes

6. Conclusions and future work

In this paper, we presented a comprehensive performance evaluation of different established machine learning algorithms used to automatically identify spam hosts based on features extracted from their web pages. For this, we employed two real, public and large datasets composed by samples represented by content-based, link-based, transformed link-based features and their combinations.

In general, the bagging of decision trees achieved the best overall results in the scenarios in which we did not combine features. In experiments with combination of feature vectors, the AdaBoost achieved the highest performance. However, through a statistical analysis we note that, in the experiment with WEBSPAM-UK2006 dataset with balanced classes, the random forest and the MLP neural networks achieved results statistically equal to the ones achieved by the bagging and AdaBoost. For WEBSPAM-UK2007 dataset, the MLP-LM and random forest also stayed in the group of the top five methods. This shows that they are promising techniques for web spam detection.

The results also indicated that all the evaluated techniques are superior when trained with balanced classes. Therefore, we can conclude that the learning algorithms tend to be biased to the benefit of the majority class.

For future work, we intend to create new feature sets composed by the fusion of the three types of features adopted in this work. We also aim to propose new approaches to combine the predictions achieved by the learning algorithms trained with each feature set.

Acknowledgment

The authors would like to thank the financial support of Brazilian agencies FAPESP, Capes and CNPq.

References

[1] Z. Gyongyi and H. Garcia-Molina, "Spam: It's not just for inboxes anymore," *Computer*, vol. 38, no. 10, pp. 28–34, 2005.

[2] K. M. Svore, Q. Wu, and C. J. Burges, "Improving web spam classification using rank-time features," in *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb'07)*, Banff, Alberta, Canada, 2007, pp. 9–16.

[3] G. Shen, B. Gao, T. Liu, G. Feng, S. Song, and H. Li, "Detecting link spam using temporal information," in *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM'06)*, Hong Kong, China, 2006, pp. 1049–1053.

[4] M. Egele, C. Kolbitsch, and C. Platzer, "Removing web spam links from search engine results," *Journal in Computer Virology*, vol. 7, pp. 51–62, 2011.

[5] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, "deSEO: combating search-result poisoning," in *Proceedings of the 20th USENIX conference on Security (SEC'11)*, Berkeley, CA, USA, 2011, pp. 20–20.

[6] L. Lu, R. Perdisci, and W. Lee, "SURF: detecting and measuring search poisoning," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, New York, NY, USA, 2011, pp. 467–476.

[7] R. M. Silva, T. A. Almeida, and A. Yamakami, "Artificial neural networks for content-based web spam detection," in *Proc. of the 14th International Conference on Artificial Intelligence (ICAI'12)*, Las Vegas, NV, USA, 2012, pp. 209–215.

[8] ——, "Towards web spam filtering with neural-based approaches," in *Advances in Artificial Intelligence – IBERAMIA 2012*, ser. Lecture Notes in Computer Science, vol. 7637. Cartagena de Indias, Colombia: Springer Berlin Heidelberg, 2012, pp. 199–209.

[9] ——, "An analysis of machine learning methods for spam host detection," in *Proc. of the 11th International Conference on Machine Learning and Applications (ICMLA'12)*, Boca Raton, FL, USA, 2012, pp. 227–232.

[10] J. Lin, "Detection of cloaked web spam by using tag-based methods," *Expert Systems with Applications: An International Journal*, vol. 36, no. 4, pp. 7493–7499, 2009.

[11] A. V. Sunil and A. Sardana, "A reputation based detection technique to cloaked web spam," *Procedia Technology*, vol. 4, no. 0, pp. 566–572, 2012.

[12] N. Spirin and J. Han, "Survey on web spam detection: principles and algorithms," *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 2, pp. 50–64, 2012.

[13] M. Najork, "Web spam detection," in *Encyclopedia of Database Systems*. Springer US, 2009, vol. 1, pp. 3520–3523.

[14] M. R. Henzinger, R. Motwani, and C. Silverstein, "Challenges in web search engines," *SIGIR Forum*, vol. 36, no. 2, pp. 11–22, 2002.

[15] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, McLachlan, A. Ng, B. Liu, P. S. Yu, Z. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. New York, NY, USA: Prentice Hall, 1998.

[17] C. M. Bishop, *Neural Networks for Pattern Recognition*, 1st ed. Oxford, UK: Oxford Press, 1995.

[18] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[19] T. Kohonen, "The self-organizing map," in *Proceedings of the IEEE*, vol. 9, no. 78, 1990, pp. 1464–1480.

[20] M. J. L. Orr, "Introduction to radial basis function networks," 1996.

[21] C. Cortes and V. N. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[22] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[23] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," National Taiwan University, Tech. Rep., 2003.

[24] J. R. Quinlan, *C4.5: programs for machine learning*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[25] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[27] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.

[28] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*. Bari, Italy: Morgan Kaufmann, 1996, pp. 148–156.

[29] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[30] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 1998.

[31] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine Learning*, vol. 11, no. 1, pp. 63–90, 1993.

[32] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, Montreal, Quebec, Canada, 1995, pp. 338–345.

[33] C. Castillo, D. Donato, and A. Gionis, "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, Amsterdam, The Netherlands, 2007, pp. 423–430.

[34] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates, "Using rank propagation and probabilistic counting for link-based spam detection," in *Proceedings of the 2006 Workshop on Web Mining and Web Usage Analysis (WebKDD'06)*, Philadelphia,USA, 2006.

[35] J. Shao, "Linear model selection by cross-validation," *Journal of the American Statistical Association*, vol. 88, no. 422, pp. 486–494, 1993.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[37] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[38] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 4, pp. 463–484, 2012.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[40] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 3rd ed. New York, NY, USA: John Wiley & Sons, 2002.