Expert Systems with Applications 42 (2015) 1166–1177

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Comparisons of machine learning techniques for detecting malicious webpages

H.B. Kazemian *, S. Ahmed

Intelligent Systems Research Centre, School of Computing, London Metropolitan University, United Kingdom

ARTICLE INFO

Article history:
Available online 16 September 2014

Keywords:
K-Nearest Neighbor
Support Vector Machine
Naive Bayes
Affinity Propagation
K-Means
Supervised and unsupervised learning

ABSTRACT

This paper compares machine learning techniques for detecting malicious webpages. The conventional method of detecting malicious webpages is going through the black list and checking whether the webpages are listed. Black list is a list of webpages which are classified as malicious from a user's point of view. These black lists are created by trusted organizations and volunteers. They are then used by modern web browsers such as Chrome, Firefox, Internet Explorer, etc. However, black list is ineffective because of the frequent-changing nature of webpages, growing numbers of webpages that pose scalability issues and the crawlers' inability to visit intranet webpages that require computer operators to log in as authenticated users. In this paper therefore alternative and novel approaches are used by applying machine learning algorithms to detect malicious webpages. In this paper three supervised machine learning techniques such as K-Nearest Neighbor, Support Vector Machine and Naive Bayes Classifier, and two unsupervised machine learning techniques such as K-Means and Affinity Propagation are employed. Please note that K-Means and Affinity Propagation have not been applied to detection of malicious webpages by other researchers. All these machine learning techniques have been used to build predictive models to analyze large number of malicious and safe webpages. These webpages were downloaded by a concurrent crawler taking advantage of gevent. The webpages were parsed and various features such as content, URL and screenshot of webpages were extracted to feed into the machine learning models. Computer simulation results have produced an accuracy of up to 98% for the supervised techniques and silhouette coefficient of close to 0.96 for the unsupervised techniques. These predictive models have been applied in a practical context whereby Google Chrome can harness the predictive capabilities of the classifiers that have the advantages of both the lightweight and the heavyweight classifiers.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Web security threats are increasing day by day (Facebook, 2010; Malware, 2011; Sood & Enbody, 2011). The open nature of the Internet allows malicious webpages to pose as 'safe webpages' and consequently some users are misled to think that these webpages are safe.

As the use and the speed of the Internet increased over the last two decades, web developers have increased the usage of images, JavaScript and other elements. The Google search engine is a clear example. At the beginning, it had very few elements. There are now more elements, graphics, stylesheets and the HTML specifications which have been added as time went on. Initially, the only way to create a webpage was by static HTML. JavaScript was then added for user interactivity. ActiveX, Silverlight, Java Applets, etc. were further added to include features. For example, ActiveX allowed browsers to host various executables which enabled users to read PDF and various other file formats such as Flash, DivX, etc. Web developers started using the integrated development environments that generated a considerable HTML markup language and this increased the HTML payload. The number of browsers increased and some of these browsers, especially Internet Explorer had their own quirks and needed more work from the developers. These factors raised the complexity of the webpages that led to potential increase in how webpages are 'adversely affected' and have become malicious.

Cross Site Scripting (XSS) injects malicious code from an unexpected source. These malicious codes can get hold of the cookies, browsing history and then send them over to the malicious webpage. Thus the user's privacy is jeopardized. There have been many attempts to prevent this sort of attacks (Lucca, Fasolino, Mastoianni, & Tramontana, 2004). XSS not only affects the user but also it affects the server. The webpage is used as the vehicle to transfer infections to multiple users. The malicious

* Corresponding author.

http://dx.doi.org/10.1016/j.eswa.2014.08.046
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

code then executes in the user's browser. The problem has been intensified with the addition of scripting capabilities that did not exist at the beginning of the history of web browsing. With the addition of scripting capabilities, the users are benefitting with a better user experience but have become prone to these additional security problems. These scripts run on users' browsers. The web developer may build a webpage only using HTML, but an attacker can still inject scripts to make it susceptible to scripts. These scripts can then access the cookies that are used for authentication. The XSS vulnerability therefore affects the users and the webpages. Take for example, a user visits a webpage and decides to purchase a product. The user adds the items to the basket and would like to checkout. Then he fills in a form to register. Each of these users is uniquely identifiable by the webpage through the use of cookies. The criminal will be able to look at the cookies and impersonate the user and buy the products, without the knowledge of the user. By the time the user has realized the problem, the money has already been dispatched from the user's account.

Almost all HTML tags are wrapped by 'greater than' and 'less than'. To write the script tag, these two characters are needed. There are several combinations of characters that can be generated (Braganza, 2006). The combinations are quite vast and will likely to increase. The combinations of letters that generate the letters are dependent on browser version and the default language. Braganza (2006) states that the browser cannot be trusted because of these extensive possibilities and some precautions are required. To cleanse, data entered are encoded and data displayed are both decoded, this process is known as 'sanitization'. In terms of how the webpage is deployed to the user, the operations team have to make sure that the firewall or any other forms of preventative measures are kept up to date. Another security threat that is very difficult to detect is clickjacking. This is a relatively new threat that has become more prevalent through the advancement of modern browsers. The interesting thing about clickjacking is that it does not use security vulnerabilities, rather uses the browsers most common feature such as hyperlinks. The user is encouraged to click a link to a webpage. But this webpage has two webpages one is displayed to the user and the other one the malicious webpage which is hidden from the user (Hansen & Grossman, 2008). The hidden webpage executes the malicious code even though the user thinks that the information is on the right webpage. This technique is very hard to detect by inspecting the source code and there have not been many successful ways to prevent it from happening.

Drive-by-download occurs without the knowledge of a user and the downloaded file is used for malicious purposes. This malicious executable installs itself on users' computer. This is a very popular method that has been used by Harley and Bureau (2008) to spread malware infection on the Internet. There are three components in the attack, the web server, the browser and the malware. An attacker finds a web server to serve the malware. The user who visits a webpage hosted in this web server is then exploited by the webpage, and some code utilizes software loopholes to execute commands on the user's browser are injected. The web server subsequently provides the malware that is downloaded by the browser. As a result, the browser that is targeted will have a known vulnerability that the attacker will try to exploit. Internet Explorer had many instances of ActiveX loopholes that the attackers had used and are still using and Harley and Bureau (2008) have provided potential solutions. The first solution is to completely isolate the browser from the operating system so that the arbitrary codes are not at all executed on the browser. Another solution is for web crawlers to visit webpage and see whether they are hosting any malware content. But the attackers can avoid by using a URL that does not have a corresponding hyperlink. Crawlers by its nature only visit URLs that have a corresponding hyperlink.

Browsers these days use publicly available blacklists of malicious webpages. These blacklists are updated after a few days or even a month. These gaps allow for webpages to be affected while being unnoticed to the crawler. At this point, the users will also get affected, because the browser thinks the webpage to be secure, as it has never been in the blacklist. Take another scenario where a regular webpage may be hacked and injected with malicious code visible only to some users or a group of users of an organization or a country. The blacklists will not be able to 'blacklist' those either. Some crawlers do not validate the JavaScript code because the code is executed on the server and not in a browser. This allows client vulnerabilities to pass through easily. Even though some of the scripts which are assumed to be safe, these scripts can load malicious scripts remotely and then execute them on the computer. Some scripts create iFrames and then load external webpages that are malicious (Provos, Mavrommatis, Rajab, & Monrose, 2008). These external webpages then gets hold of the cookies and steal the identity. The users then browse this malicious webpage and get infected and are then easily tracked by remote users from somewhere else. The users also may run malicious executables without even knowing that the executables have already access to the system and are monitored from somewhere else. Webpages are the common victims to all these threats that have been described above. The features in a webpage can indicate whether it is malicious or not. Researchers have studied and analyzed a large number of features with or without machine learning techniques described below.

Kan and Thi (2005) carried out one of the first research work that utilized machine learning to detect malicious webpages. This work ignored webpage content and looked at URLs using a bag-of-words representation of tokens with annotations about the tokens' positions within the URL. A noteworthy result from Kan and Thi's research is that lexical features can achieve 95% accuracy of the page content features. Garera, Provos, Chew, and Rubin (2007)s work used logistic regression over 18 hand selected features to classify phishing URLs. The features include the presence of red flag key words in the URL, features based on Google's page ranking, and Google's webpage quality guidelines. Garera et al. achieved a classification accuracy of 97.3% over a set of 2500 URLs. Although this paper has similar motivation and methodology, it differs by trying to detect all types of malicious activities. This paper also uses more data for training and testing, as described in the subsequent sections. Spertus (1997) suggested an alternative approach and endeavored to identify malicious webpages, Cohen (1996) employed the decision trees for detection and Dumais, Platt, Heckerman, and Sahami (1998) utilized inductive learning algorithms and representations for text categorization. Guan, Chen, and Lin (2009) focused on classifying URLs that appear in webpages. Several URL-based features were used such as webpage timing and content. This paper has used similar techniques but applied them to webpages which have much more complex structures with better accuracies. Mcgrath and Gupta (2008) did not construct a classifier but performed a comparative analysis of phishing and non-phishing URLs. With respect to data sets, they compared non-phishing URLs drawn from the DMOZ Open Directory Project to phishing URLs from Phishtank (2013) and a non-public source. The features they analyzed included IP addresses, WHOIS thin records (containing date and registrar provided information only), geographic information, and lexical features of the URL (length, character distribution and presence of predefined brand names). The difference is that this paper utilizes different types of features to add to the novelty. Provos et al. (2008) carried out a study of drive-by exploit URLs and used a patented machine learning algorithm as a pre-filter for virtual machine (VM) based analysis. This approach is based on heavyweight classifiers and is time consuming. Provos et al. (2008) used the following features in computer simulation,

content based features from the page, whether inline frames are 'out of place', the presence of obfuscated JavaScript, and finally whether iFrames point to known malicious sites. Please note, an 'iFrame' is a window within a page that can contain another page. In their evaluations, the machine learning based pre filter can achieve 0.1% false positives and 10% false negatives. Provos et al.'s approach is very different to this paper as the features are primarily focused on iFrames. Bannur, Saul, and Savage (2011)s research has some similarities to this paper but it uses a very small dataset and furthermore this paper utilizes various other types of features. Abbasi, Zhang, Zimbra, Chen, and Nunamaker (2010) and Abbasi, Zahedi, and Kaza (2012) ran some classification to detect fake medical sites but the size of dataset was very small, whereas this paper focuses on detecting any type of malicious webpages. Fu, Wenyin, and Deng (2006) and Liu, Deng, Huang, and Fu (2006)s research considered the visual aspects of a webpage to determine whether the page is malicious or not. Le, Markopoulou, and Faloutsos (2010) detected phishing webpages only using the URLs. And, Ma, Saul, Savage, and Voelker (2009b) looked at online learning to detect malicious webpages from URL features.

As the security improves, the attackers will devise new ways to avoid barriers raised by administrators and web developers. To further improve security, an automated tool is required in order to detect the vulnerabilities. One approach to automation is for web developers to secure and enhance their webpages. But there are limits to the extent that developers can work to secure webpages. Web developers are bound by the web frameworks they use (Okanovic, Mateljan, & May, 2011). If the web frameworks fail to take preventative measures, the users' machines get infected and the webpages become vulnerable. This paper takes the research in malicious webpages described above further by applying three supervised machine learning techniques such as Naive Bayes Classifier, K-Nearest Neighbor and Support Vector Machine, and two unsupervised machine learning techniques like K-Means and Affinity Propagation, and compares the results. The novel unsupervised techniques of K-Means and Affinity Propagation have not been applied to detection of malicious webpages by any other researchers in the past. Moreover, to add to the novelty, the research utilizes more complex structures for better accuracies, more data for training and testing, and employs both the lightweight and the heavyweight classifiers for detection of all types of malicious activities.

2. Machine learning models

This section provides a brief overview of the five machine learning techniques that have been used in this paper; they are Naive Bayes Classifier, K-Nearest Neighbor, Support Vector Machine, K-Means and Affinity Propagation, briefly described below. The computer simulation results of these machine learning methods are discussed in Section 3.

2.1. Naive Bayes Classifier

Naive Bayes Classifier uses the Bayes' theorem. The classifier assumes that all the features are independent of each other. It learns $p(\mathscr{C}_k|x)$ by modeling $p(x|\mathscr{C}_k)$ and $p(\mathscr{C}_k)$, using Bayes' rule to infer the class conditional probability (Bayes & Price, 1763). The model is

$$\begin{aligned} y(\mathbf{x}) &= \arg\max_x p(\mathscr{C}_k|x) = \arg\max_x p(x|\mathscr{C}_k) \times p(\mathscr{C}_k) \\ &= \arg\max_x \prod_{i=1}^{D} p(x_i|\mathscr{C}_k) \times p(\mathscr{C}_k) \\ &= \arg\max_x \sum_{i=1}^{D} \log p(x_i|\mathscr{C}_k) + \log p(\mathscr{C}_k) \end{aligned} \tag{1}$$

The training in this paper was carried out using Gaussian likelihood (alternative options for training include multivariate likelihood and multinomial likelihood).

$$p(x|\mathscr{C}_k) = \prod_{i=1}^{D} \mathscr{N}(\mu_{ik}, \sigma_{ik}) \tag{2}$$

where:

- C_k are the classes where $C = \{C_1, C_2, \ldots, C_k\}$.
- $\mathscr{N}(\mu_{ik}, \sigma_{ik})$ is the normal distribution.
- μ is the mean of the Gaussian distribution.
- σ is the standard deviation of the Gaussian distribution.

The complexity of the model $\mathscr{O}(NM)$ is as such that each training instance must be visited and each of its features ought to be counted. For non-linear problems, it can only learn linear boundaries for multivariate/multinomial attributes. With Gaussian attributes, quadratic boundaries can be learnt with unimodal distributions (Jordan, 2002). Naive Bayes is generally used in text classification and is one of the most widely used classification algorithm in machine learning because it is fast and space efficient, which is also noticed in the simulation results described in Section 3.

2.2. K-Nearest Neighbor

K-Nearest Neighbor works in such a way that the label of a new point $\hat{x}$ is classified with the most frequent label $\hat{t}$ of the k nearest training instances. It is modeled as

$$\hat{t} = \arg\max_{\mathscr{C}} \sum_{i:x_i \in N_k(\mathbf{x},\hat{x})} \delta(t_i, \mathscr{C}) \tag{3}$$

where:

- $N_k(\mathbf{x}, \hat{x}) \leftarrow k$ points in $\mathbf{x}$ closest to $\hat{x}$.
- Euclidean distance formula: $\sqrt{\sum_{i=1}^{D}(x_i - \hat{x}_i)^2}$.
- $\delta(a, b) \leftarrow 1$ if $a = b$; 0o/w.

The model does not require any optimization and it trains itself using cross validation to learn the appropriate k. k regularizes the classifier, as $k \to N$ the boundary becomes smoother. $\mathscr{O}(NM)$ is used as space complexity, since all training instances and all their features need to be kept in memory. K-Nearest Neighbor uses a very simple technique for classification, and cannot handle large training dataset as shown in the results section. It can handle non-linear boundaries easily (Altman, 1992).

2.3. Support Vector Machines

SVM was developed by Cortes and Vapnik (1995) and it is widely regarded as one of the most effective models for binary classification of high dimensional data. SVM and indeed any other supervised classifiers use a similar technique to classify webpages shown in Fig. 1. SVM is modeled as

$$h(x) = b + \sum_{n=1}^{N} y_i \alpha_i K(x, x_i) \tag{4}$$

where

- $h(x)$ is the distance of the decision boundary.
- b is the bias weight.
- α is a coefficient that maximize the margin of correct classification on the training set.
- N is the number of features.
- K is the kernel function.
- x is a feature vector.

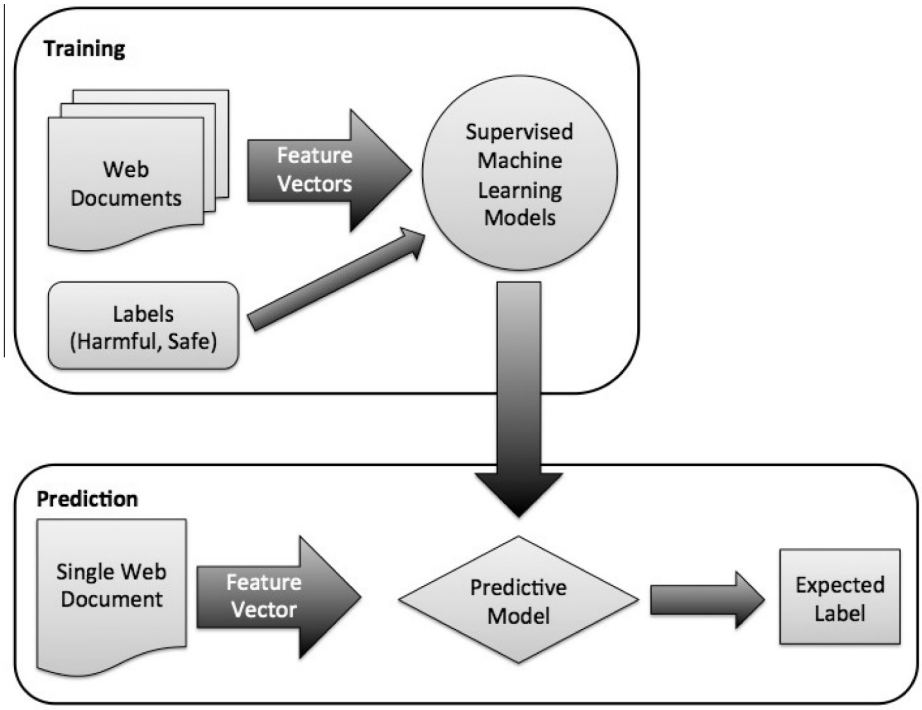

Fig. 1. Supervised machine learning architecture for classifying webpages.

The positive or negative sign of this distance indicates the side of the decision boundary on which the example lies. The value of $h(x)$ is limited to predict a binary label for the feature vector x. The model is trained by initially specifying a kernel function $K(x, x')$ and then computing the coefficients α_i that maximize the margin of correct classification on the training set. The required optimization can be formulated as an instance of quadratic programming, a problem for which many efficient solutions have been developed (Chang & Lin, 2012).

2.4. K-Means

K-Means is a geometric clustering, hard-margin algorithm, where each data point is assigned to its closest centroid. It is modeled using hard assignments $r_{nk} \in \{0, 1\}$ so that $\forall n \sum_k r_{nk} = 1$, i.e. each data point is assigned to one cluster k (MacQueen, 1967). The geometric distance is calculated using the Euclidean distance, l^2 norm:

$$||x_n - \mu_k||^2 = \sqrt{\sum_{i=1}^{D} (x_{ni} - \mu_{ki})^2} \tag{5}$$

where

- μ_k is cluster centroid.
- D is the no of points.
- x is one of the points.

The mini-batch K-Means algorithm uses mini batches to reduce the computation time while still attempting to optimize the same objective function. Mini-batches are subsets of the input data, randomly sampled in each training iteration. These mini-batches drastically reduce the amount of computation required to converge to a local solution. Mini batch k-means converges faster than K-Means, but the quality of the results is reduced. In practice, the difference in quality can be quite small, as shown in the results section.

2.5. Affinity Propagation

Affinity Propagation is an unsupervised algorithm created by Dueck (2009) where each data point acts as centroids and these data points choose the number of clusters. The following is an example of how to represent the centroid for datapoint i

$$c_i \in \{1, \ldots, N\} \tag{6}$$

The algorithm maximizes the following function

$$S(\mathbf{c}) = \sum_{i=1}^{N} s(i, c_i) + \sum_{k=1}^{N} \delta_k(\mathbf{c}) \tag{7}$$

The similarity of each point to the centroid is measured by the first term in Eq. (7) and the second term is a penalty term denoted as $-\infty$. If some data point i has chosen k as its exemplar that is $c_k \neq k$, but k has not chosen itself as an exemplar i.e. $c_k = k$, then the following constraints could be presented in Eq. (8).

$$\delta_k(\mathbf{c}) = \begin{cases} -\infty & \text{if} \quad c_k \neq k \text{ but } \exists_i : c_i = k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

A factor graph can represent the objective function and it is possible to use N nodes, each with N possible values or with N^2 binary nodes. Each variable node c_i sends a message to each feature node δ_k and each factor node δ_k sends a message to each variable node c_i. The number of clusters is controllable by scaling the diagonal term $S(i,i)$, which shows how much each data point would like to be an exemplar. Although Affinity Propagation was developed very recently it has a very good performance as shown later in the results section. The overall architecture of the unsupervised machine learning is outlined in Fig. 2.

3. Results

The architecture of the computer simulation carried out for this paper is presented in Fig. 3. The experiment was conducted on an Intel Xeon E3-1220 4 Cores × 3.1 GHz computer with 12 GB of RAM. First, 100,000 webpages were downloaded using a crawler

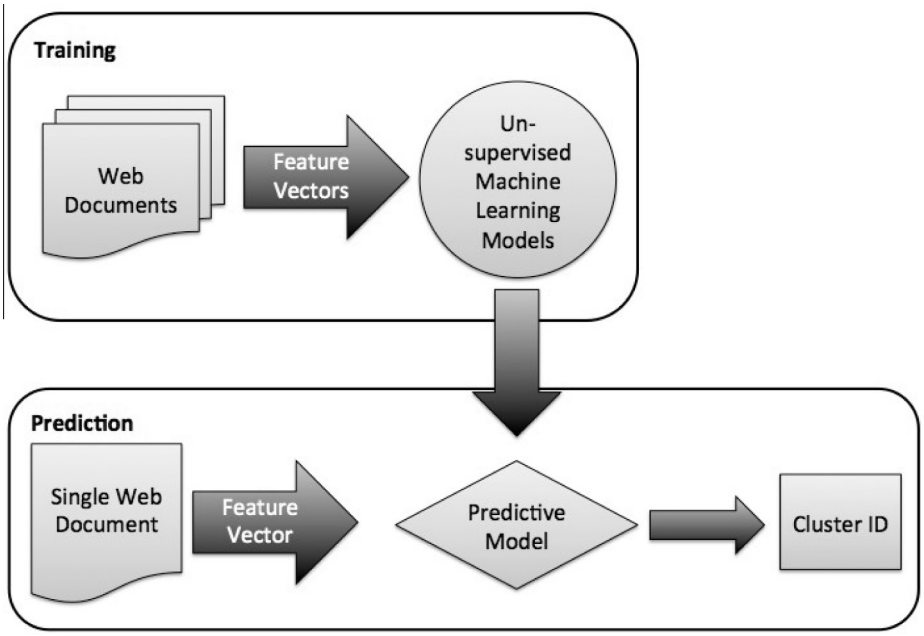

Fig. 2. Unsupervised machine learning architecture for classifying webpages.

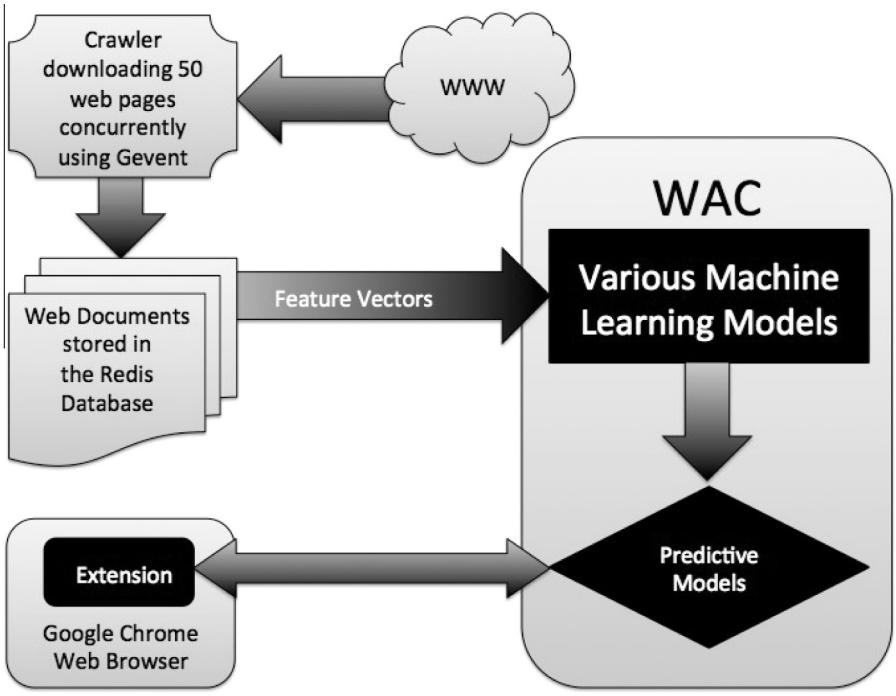

Fig. 3. The architecture of Web Application Classifier (WAC).

and then converted into feature vectors. Then a tool named Web Application Classifier (WAC) took these vectors as inputs, and applied the machine learning algorithms described in the previous section to create the predictive models. These predictive models read vectors of the new webpages to produce an output that indicated whether a webpage is safe or not. The Sections 3.1–3.5 describe which features were used and how the webpages were preprocessed and cleansed before placing inside the predictive models.

3.1. Data sources

The downloaded webpages were divided into two sets, malicious and safe. The source for the list of safe webpages was gathered primarily from Alexa (2013), the malicious ones were collated primarily from Phishtank (2013) and two types of representation of each webpage were created. One contained all the HTML codes and other only had English characters.

3.2. Features

3.2.1. Semantic

This paper utilizes vector space representation to extract the semantic features, as it is commonly used in classification and clustering. Salton, Wong, and Yang (1975) carried out research using a vector space model for automatic indexing of webpages, Cohen and Singer (1999) used a context-sensitive learning method

for categorizing text, and Sahami, Dumais, Heckerman, and Horvitz (1998) utilized a Bayesian approach to classify emails for detecting spam. Vector space representation denotes webpages as vectors in a very high dimensional space. Each webpage is represented as a Boolean or numerical vector; in this paper, it is represented as a numerical vector. In this very high dimensional space, each term is a sequence of alphanumeric characters. A given webpage has in each component a numerical value specifying some function f of often a term corresponding to the dimension appearing in the webpage. Salton and Buckley (1988) used an alternative term called weighting but it is not used in this simulation. In a vector space representation of webpages, $\xi(t_i,d)$ is the number of occurrences of term t_i in webpage d; $\xi(t_i,d)$ is a random variable. The function f is applied to $\xi(t_i,d)$ and produces a value for the ith component of the vector for webpage d. The identity function $f(\alpha) = \alpha$ is applied to the term counts. Other common functions that are applied to the term frequencies are outlined below.

$$f(\alpha) = log(\alpha + 1) \tag{9}$$

$$f(\alpha) = \sqrt{\alpha} \tag{10}$$

$$f(\alpha) = \frac{a}{a + const} \tag{11}$$

Eq. (9) was defined by Robertson and Jones (1976). Eq. (10) was used in the scatter/gather system (Cutting, Karger, Pedersen, & Tukey, 1992) for webpage clustering and was found to outperform Eq. (9). Robertson and Walker (1994) proposed Eq. (11) and found this general form to be useful for webpage retrieval by making use of various instantiations of the constant value.

Term frequency – inverse document frequency (TFIDF) weighting is the most used function for the webpage term frequencies. The term frequencies (TF) in each webpage and the inverse document (webpage) frequency (IDF) of each term in the entire collection are part of the weighting function. IDF is defined as

$$\text{IDF}(t) = log\left(\frac{N}{n_t}\right) \tag{12}$$

where N is the total number of webpages in the collection and n_t is the number of webpages in which term t appears at least once. The TFIDF weight for a term t in a webpage d is the product of the term frequency and the inverse webpages frequency for that term, returning:

$$\text{TFIDF}(t,d) = \xi(t,d).\text{IDF}(t) \tag{13}$$

This paper uses a simple Boolean representation of webpages that records whether or not a given term appears in a webpage. Most rule-based methods (Apté, Damerau, & Weiss 1994; Cohen, 1996) use an underlying Boolean model and Boolean vector representation has been used in probabilistic classification models. Another way to incorporate word frequency information into probabilistic classification models is by using a parametric distribution, such as bounded Gaussian or Poisson distribution to capture the probability of words appearing number of times in webpages. Yang and Chute (1994) provide further evidence that Boolean representation is adequate by comparing Boolean and frequency-based representations.

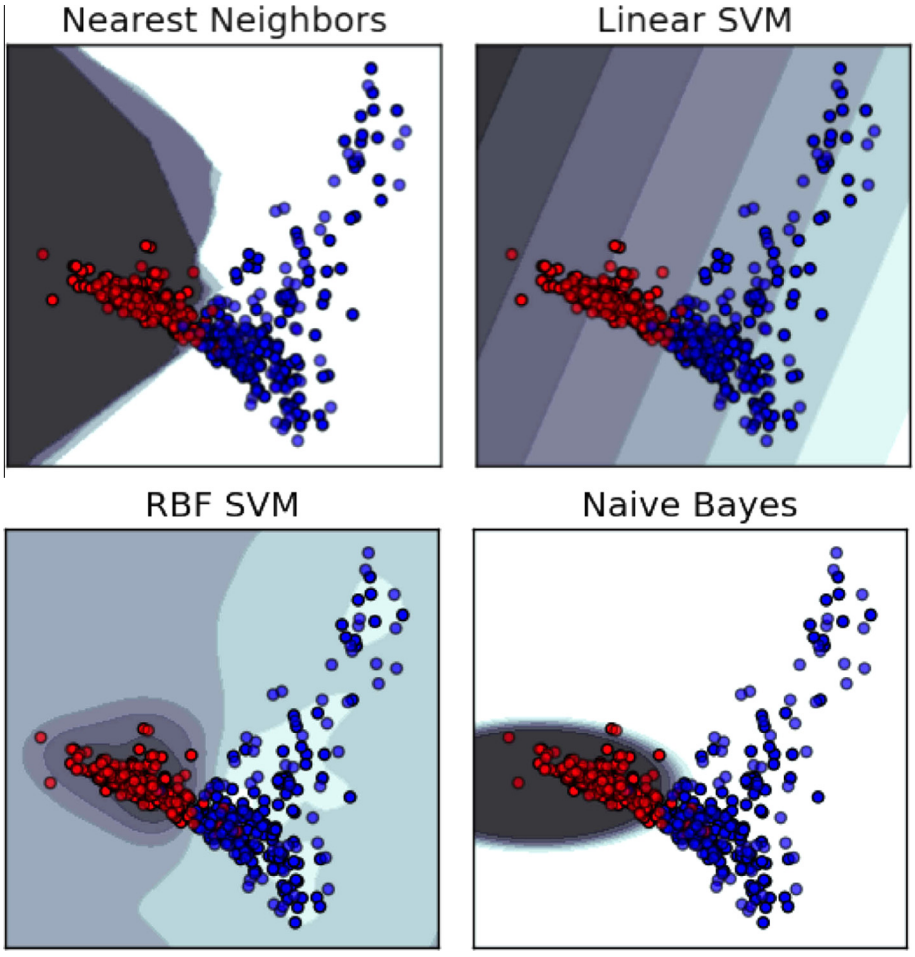

Fig. 4. Visual representation of supervised learning models demonstrating clear separations of malicious and safe webpages.

Table 1
Results of comparisons of supervised machine learning models that detect malicious webpages (key: KNN: K-Nearest Neighbor, LS: Linear SVM, RS: RBF SVM, NB: Naive Bayes).

No. of webpages	KNN (%)	LS (%)	RS (%)	NB (%)
50	74	80	79	77
100	75	82	83	78
500	79	86	92	78
5000	91	93	97	84
100,000	95	93	98	89

Word stemming reduces the words in the webpages to their root forms, known as word stems. Porter (1997) described a simulation model that utilizes the word-stemming algorithm, which is able to combine similar and dissimilar items into one. But Frakes and Baeza-Yates (1992) compared various stemming methods to unstemmed representations and showed that in many cases the performances of both representations are very close to each other.

3.2.2. URLs

URLs identify webpages and are used as unique identifiers. Many malicious webpages have suspiciously looking characters in their URLs and in their contents. Sometimes URLs have spelling mistakes too. The lexical features of URLs were fed into the machine learning models. If there were spelling mistakes or suspicious characters in the URL, then they were regarded as suspicious.

3.2.3. Page links

Webpages have many links in order to provide further information. The webpages that link to malicious webpages are likely to be malicious themselves. The computer simulations extracted all the links from each webpage and they were also fed into the machine learning models.

3.2.4. Visual features

All the features that have been mentioned so far are text based such as source codes, stripped HTML, domain names and URLs. The visual features are based on images. First, screenshots of webpages were downloaded by passing the URLs to PhantomJS (a headless webkit browser with JavaScript API). PhantomJS took each URL, saved a screenshot of the webpage and converted them to Portable Network Graphics (PNG) file format. Images were then converted to a format understandable by the proposed models. There are two popular techniques that are generally used such as Speeded Up Robust Features (SURF) and Scale Invariant Feature Transform (SIFT) (Lowe, 2004). The simulation used SURF, as it has less stringent licensing options compared to SIFT. The idea behind using the visual features is that malicious webpages look simpler because they are likely to have less input from designers whereas safe webpages are designed better.

An exciting feature was that the unsupervised machine learning models were used in conjunction with supervised machine learning models and each screenshot of the webpages were analyzed using SURF. The values from the SURF were clustered using the unsupervised machine learning models. These clusters were then fed into the supervised machine learning models to further improve the classification.

3.3. Evaluation of the machine learning models

Machine learning methods discussed previously were used on different combinations of features. First the webpages were classified from just content features, then other types of features were

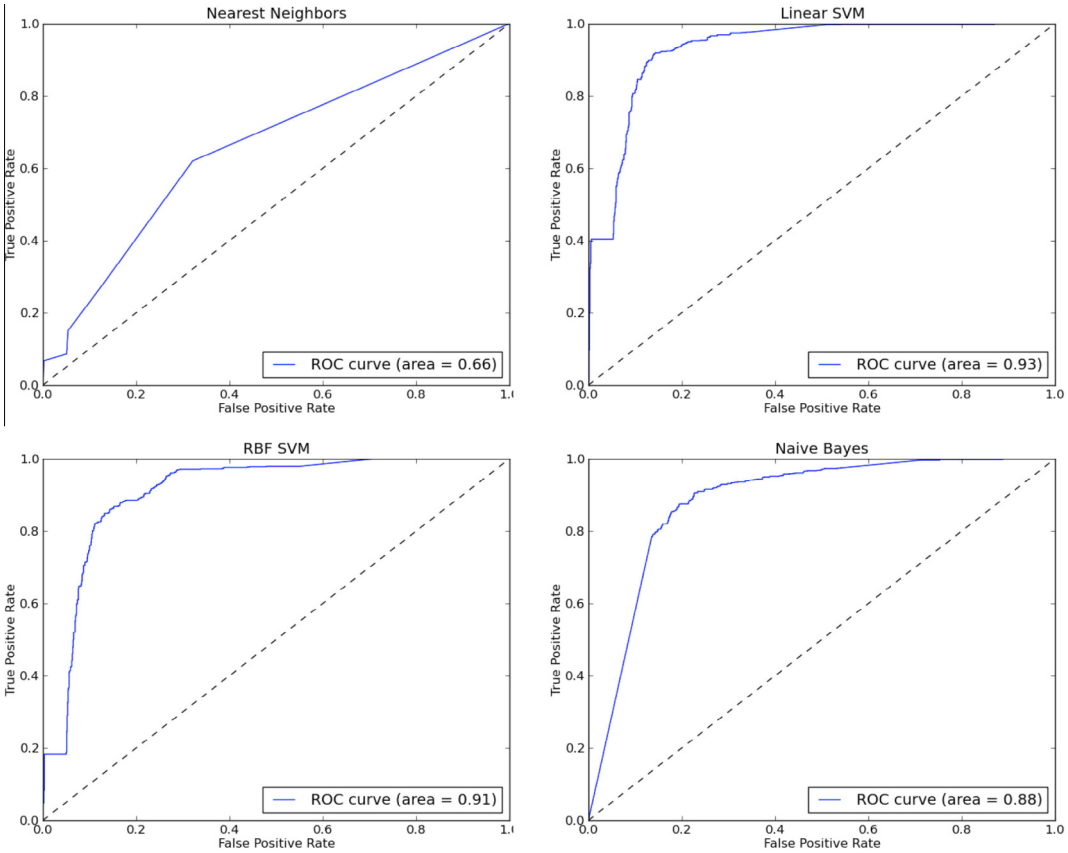

Fig. 5. ROC curves for supervised machine learning models.

added and the gain was reexamined. It was found in this research that the highest accuracy is obtained by combining URL, page-link, semantic TFIDF and SURF features, which adds to the paper's novelty. This combination of features was used as the optimal feature configuration. The accuracy of the unsupervised models had to be done differently because the truth labels are not known and the evaluation had to be performed using the model itself. Finally the machine learning techniques were trained on data sets with varying ratios of malicious and safe webpages. 70% of the labeled webpages were used for training and 30% for testing. The ratio of malicious to safe webpages is the same in testing as well as training for the supervised machine learning models.

The supervised classification performances were evaluated in terms of precision and recall, while the silhouette coefficient was used to evaluate the unsupervised techniques. The silhouette coefficient outputs a score relating to a model with better-defined clusters. The silhouette coefficient is defined for each sample and composed of two extremes, bounded between −1 for incorrect clustering and +1 for highly dense clustering.

3.3.1. Supervised techniques

Fig. 4 shows a visual representation of webpages as safe or malicious, using supervised classification. K-Nearest Neighbor, Radial Basis Function (RBF) SVM, Linear SVM and Naive Bayes clearly separate the safe and malicious webpages. The webpages used in Fig. 4 is somewhat smaller in numbers in order to demonstrate pictorial representations of the outcomes. Table 1 further demonstrates the overall results of the supervised techniques. The accuracy for all the supervised models improved as the number of webpages increased. In general, SVM outperformed the rest. In the case of SVM, the accuracy values are remarkably low even when a small number of webpages are applied. As soon as the number of webpages exceeds 500, the accuracy increases. In the case of other models, a similar trend is observed at the beginning, as the number of webpages increases the accuracy also increases. The results suggest that the models were able to generalize better as more patterns emerged from various sources. Receiver Operating

Table 2
Results based on datasets provided by Ma et al. (2009a).

Classifier	Accuracy (%)
K-Nearest Neighbor	91
RBF Support Vector Machine	97
Linear Support Vector Machine	92
Naive Bayes	85

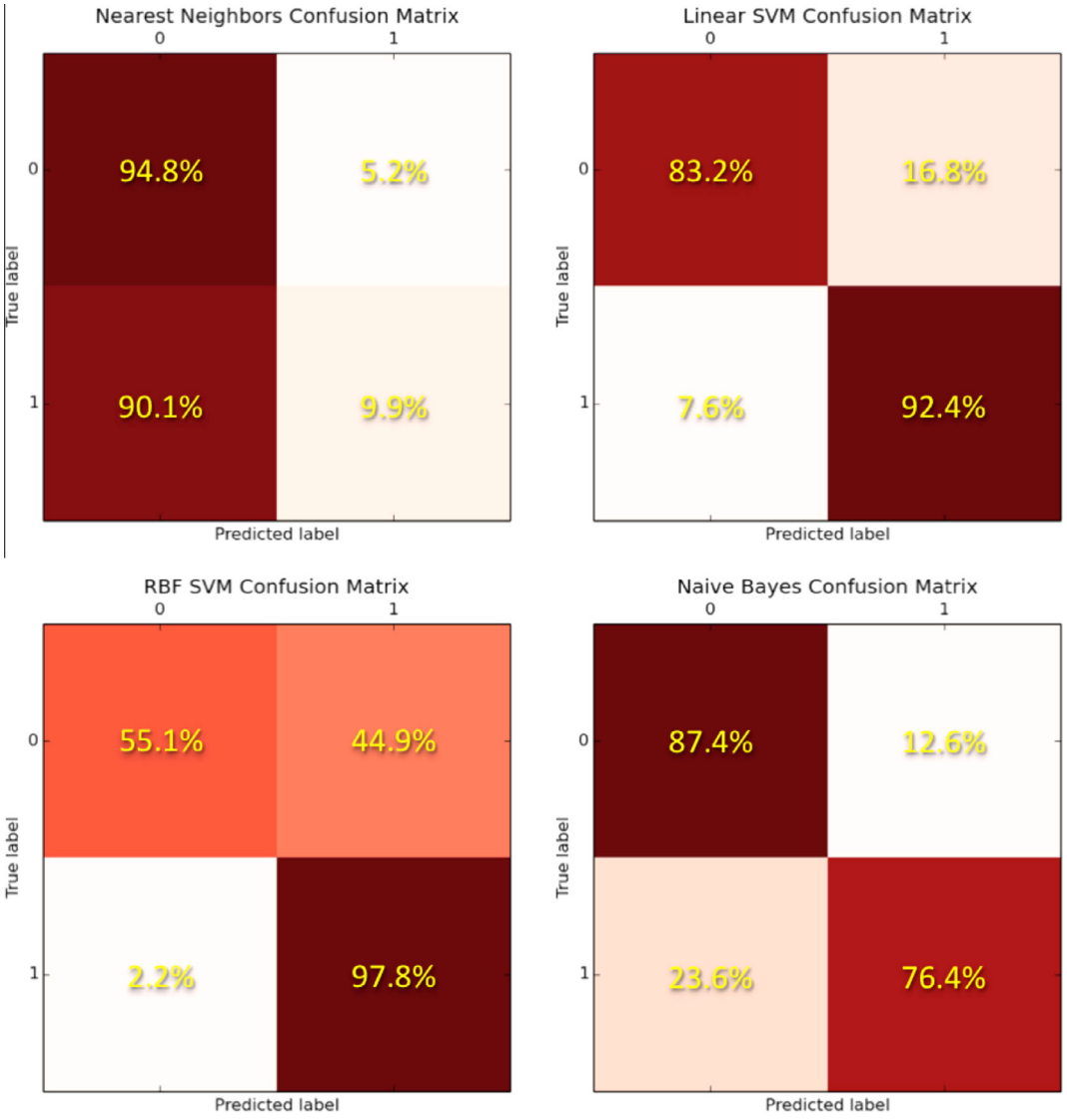

Fig. 6. Confusion matrices for supervised machine learning techniques.

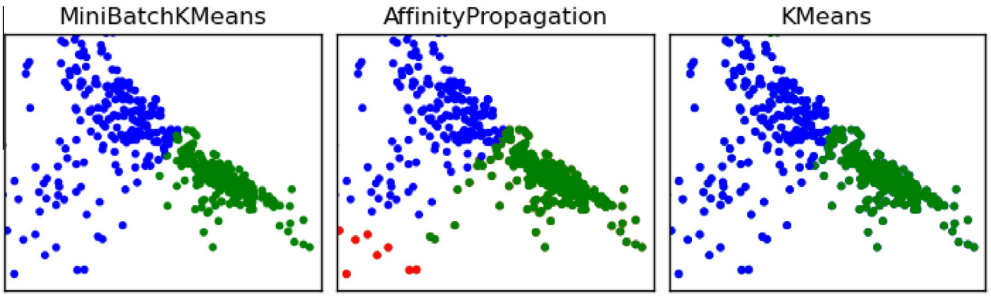

Fig. 7. Visual representation of unsupervised learning models demonstrating clear separations of malicious and safe webpages.

Table 3
Results of comparisons of unsupervised machine learning models that detect malicious webpages.

Classifier	Silhouette coefficient
Mini Batch K-Means	0.877
Affinity Propagation	0.963
K-Means	0.877

Characteristic (ROC) curve and Confusion Matrix are used to analyze the efficiencies and performances of the supervised algorithms, as outlined in Figs. 5 and 6. ROC curve is a graph which demonstrates the efficiency and performance of a classifier system by plotting true positive rate against false positive rate at various threshold settings. Fig. 5 illustrates that Linear SVM and RBF SVM perform the best out of the four classifiers with 0.93 and 0.91 respectively. K-Nearest Neighbor performs the worst, because it had less access to training data due to memory constraints. Confusion Matrix is a contingency table enables visualization of the efficiency and performance of a supervised learning algorithm, which makes it easy to understand if the system is confusing or mislabeling two classes. In Fig. 6, the confusion matrices have four sections, True Positive, True Negative, False Positive and False Negative. True Positive denotes that a malicious webpage is correctly identified as malicious. True Negative represents that a safe webpage is correctly labeled as safe. False Positive means that a safe webpage is incorrectly identified as malicious. False Negative indicates that a malicious webpage is incorrectly labeled as safe. The proposed machine learning techniques are designed as such where the priority is given to detect the malicious websites that is the true positives. With this is mind, the outstanding True Positives from highest to lowest are in the following order RBF SVM (97.8%), Linear SVM (92.4%), Naïve Bayes (76.4%) and K-Nearest Neighbor (9.9%).

The supervised models were also run against one of the most popular datasets provided by Ma, Saul, Savage, and Voelker (2009a). The data file had based on SVM and therefore the data was converted into recognizable formats to be fed into the machine learning models. Table 2 shows the results of the simulations using Ma et al. datasets. All the supervised algorithms scored over 85% and RBF SVM performed the best with 97% accuracy.

3.3.2. Unsupervised techniques

Fig. 7 shows that the three unsupervised algorithms do clearly separate the malicious and safe webpages. In Fig. 7, a smaller number of webpages were used in order to demonstrate pictorial representations of the outcomes. The detailed numerical results are outlined in Table 3 using 100,000 webpages. Affinity Propagation performs the best among the unsupervised machine learning

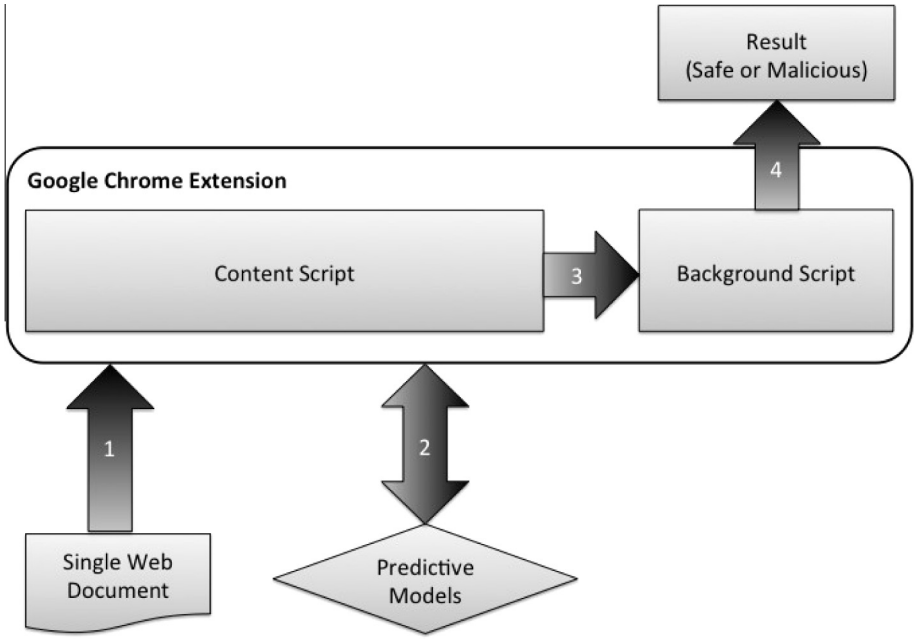

Fig. 8. The Chrome extension uses 4 steps to decide whether a webpage is malicious or not.

algorithms in Table 3 because the silhouette coefficient is closest to 1. Furthermore, Affinity Propagation algorithm identifies three clusters. The red cluster is grouped as outlier, whereas the other two Mini Batch K-Means and K-Means find only two clusters.

3.4. Online learning

The conventional batch processing machine learning models cannot learn incrementally. Online learning needs to employ a different approach than the traditional batch processing to accommodate for the new incoming data. This problem is addressed in this paper by using streaming data in the form of a list of malicious webpages and safe webpages, which are compiled from various sources. This allows the supervised predictive models inside WAC to train automatically as new the data is coming in. This has been found to be very useful with the use of Chrome extension, which is described in Section 3.5.

3.5. Chrome extension

The chrome extension has been built using the architecture shown in Fig. 8. The Content Script looks at the loading document and sends the loaded source code to the predictive classifier. The classifier then parses, creates the features and then responds with

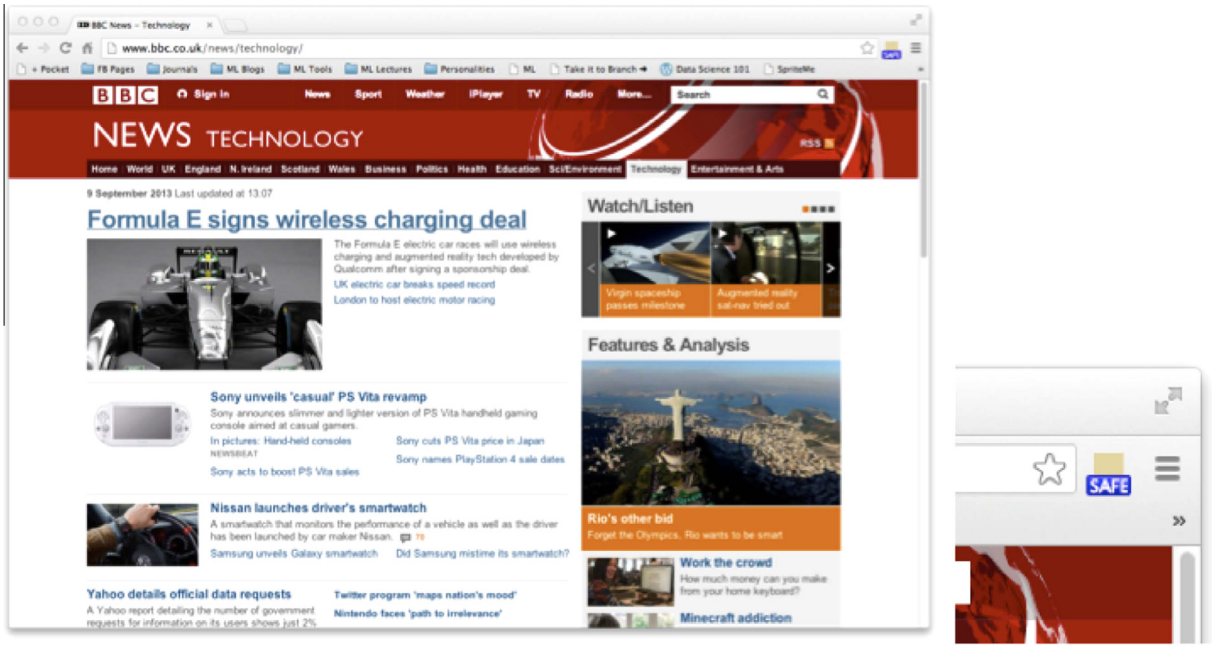

Fig. 9. The Chrome extension shows that the website loaded by the user is safe.

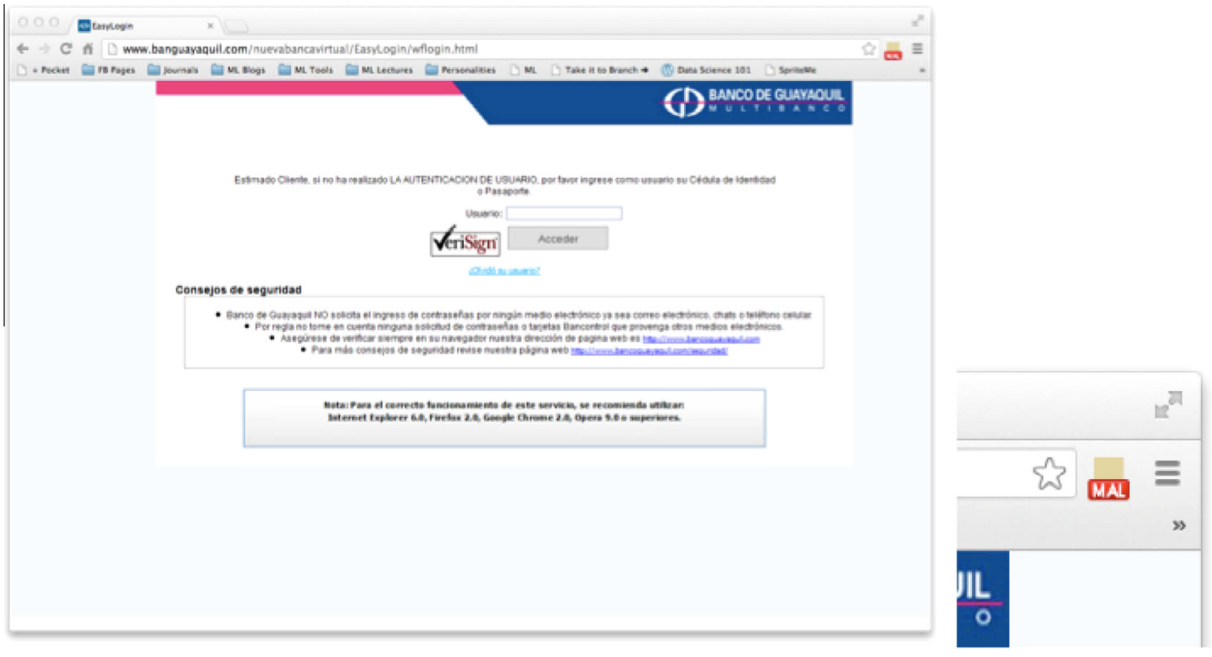

Fig. 10. The Chrome extension shows that the website loaded by the user is malicious.

whether it thinks that the webpage is safe or malicious. The Background Script receives the response and displays whether it is safe as demonstrated in Fig. 9 or malicious as shown in Fig. 10.

The four steps in the Chrome Extension are outlined below:

Step 1: Grab the webpage.
Step 2: Extract the features and send them to the predictive model. The predictive model then sends a response.
Step 3: Content Script notifies the Background Script with the response.
Step 4: Background Script displays the response.

'Heavyweight' classifiers are more accurate but they have poor prediction time. Heavyweight classifiers use more features and so have a higher accuracy. 'Lightweight' classifiers does the opposite, it uses less features and consumes the features from the browser. This paper utilizes Google Chrome which exploits the predictive capabilities of the proposed supervised and unsupervised machine learning models by taking into account the advantages of both the heavyweight and the lightweight classifiers. The Chrome extension obtains all the features from the browser and sends them to the classifier which uses more features, thus have a quick prediction time and yet higher accuracy. The Chrome extension used the supervised models for accurate classification and the unsupervised models were used to ascertain the clusters of the screenshots of webpages.

4. Conclusion

This paper presents the evaluation of three supervised machine learning models and two unsupervised machine learning models for text classification, to detect webpages as either malicious or not. All the supervised techniques were trained and applied to a large number of webpages and manually split into two classes. In a nutshell, all of the machine learning techniques show encouraging performance with accuracies above 89% for supervised techniques and a silhouette coefficient of 0.87 for unsupervised techniques using 100,000 webpages. As the number of webpages increased the accuracy rates for each type of machine learning model were improved. The accuracies of the unsupervised models are not better than the supervised one, but came close. This is interesting because the unsupervised models were not aware of the two classes of malicious and safe.

The major contribution of this paper is to explore a range of machine learning algorithms that use a wide range of features including the use of unsupervised algorithms. The computer simulation results show that the classifiers can obtain information from the URLs, page links, semantics and visual features of webpages. Different sets of data on ratios of malicious and safe webpages do not significantly affect the error rates of the classifiers. The content features of webpages contributed the most to reduce the error rates, and then the other significant ones were the URLs followed by the SURF visual features. The visual features take more computational resources and they are time consuming to compute. However, the SURF features can capture the visual information faster than the SIFT. Another significant contribution of this paper is the use of Chrome extension in the browser. This allowed to detect whether a webpage is malicious or not in a very short time. The lightweight classifiers are very fast but are slightly less accurate, whereas heavyweight classifiers are more accurate but take more time. This extension enabled the SVM classifier to be 'middleweight' with very high accuracy and yet less prediction time. The final contribution is the use of online learning to detect malicious webpages, which allows for the training to take place without going through the 'old training data'. This is very significant for real world applications. There is one limitation in this paper, that is malicious webpages will not be detected if the harmful elements are outside of the features that have been used in the proposed algorithms. Despite this limitation, the paper's contribution is encouraging as the accuracy rates are improved through the use of visual features.

There are possibilities that this research work could be taken further, some of which are described below. (i) Changing the implementation of the whole system to JavaScript which will combine the separate processes into one. This approach will allow the application to run on the browser as a standalone extension without external dependencies. This method requires that the machine learning models to be rewritten. The question to be answered is whether the prediction time will still remain the same. (ii) Using deep learning to extract features automatically. With the current scheme all the features have to be specified in the implementation and it is difficult to incorporate new features because the system has to be pre-processed and it is time consuming. Deep learning may be able to extract features automatically and probably will adapt to the future threats that could use new features which have not been discovered as yet. The question remains the same whether deep learning will be as effective as the current successful systems. (iii) Using multiple processing units of a Graphical Processing Unit. Using distributed processing in general will improve the time to build the machine learning models and also will improve the prediction time. However, the main challenge will be to rewrite the algorithms so that they will make an efficient use of hardware especially on mobile devices.

Acknowledgement

The authors would like to thank Technology Strategy Board - KTP, UK for their generous support for part of the research [Grant No. ktp006367].

References

Abbasi, A., Zahedi, F. M., & Kaza, S. (2012). Detecting fake medical web sites using recursive trust labeling. *ACM Transactions of Information Systems, 30*(4), 22:1–22:36. URL http://dx.doi.org/10.1145/2382438.2382441.

Abbasi, A., Zhang, Z., Zimbra, D., Chen, H., & Nunamaker, J. F. (2010). Detecting fake websites: The contribution of statistical learning theory. *MIS Quarterly, 34*(3), 435–461. URL <http://dl.acm.org/citation.cfm?id=2017470.2017473>.

Alexa (2013). Alexa. URL <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>.

Altman, N. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175–185.

Apt´e, C., Damerau, F., & Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions of Information Systems, 12*(3), 233–251. URL http://dx.doi.org/10.1145/183422.183423.

Bannur, S. N., Saul, L. K., & Savage, S. (2011). Judging a site by its content: learning the textual, structural, and visual features of malicious web pages. In *Proceedings of the 4th ACM workshop on security and artificial intelligence. AISec '11* (pp. 1–10). New York, NY, USA: ACM. URL http://dx.doi.org/10.1145/2046684.2046686.

Bayes, M., & Price, M. (1763). An essay towards solving a problem in the doctrine of chances. By the late rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to john canton, A. M. F. R. S.. *Philosophical Transactions, 53*, 370–418. URL <http://rstl.royalsocietypublishing.org/content/53/370.short>.

Braganza, R. (2006). Cross-site scripting: Cross-site scripting an alternative view. *Network Security, 2006*(9), 17–20. URL http://dx.doi.org/10.1016/S1353-4858(06)70425-1.

Chang, C. -C., & Lin, C. -J. (2012). Libsvm: A library for support vector machines. URL <http://www.csie.ntu.edu.tw/cjlin/libsvm/>.

Cohen, W. W. (1996). Learning trees and rules with set-valued features. *Proceedings of the thirteenth national conference on artificial intelligence. AAAI'96* (Vol. 1, pp. 709–716). Springer.

Cohen, W., & Singer, Y. (1999). Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems (TOIS), 17*(2), 141–173.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 273–297.

Cutting, D. R., Karger, D. R., Pedersen, J. O., & Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on research and development in information retrieval. SIGIR '92* (pp. 318–329). New York, NY, USA: ACM. URL http://dx.doi.org/10.1145/133160.133214.

Dueck, D. (2009). Affinity propagation: Clustering data by passing messages.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on information and knowledge management. CIKM '98* (pp. 148–155). New York, NY, USA: ACM. URL http://dx.doi.org/10.1145/288627.288651.

Facebook (2010). Facebook suffers from rash of clickjacking. Network Security 2010 (6), 20. URL <http://www.sciencedirect.com/science/article/pii/S1353485810700866>.

Frakes, W. B., & Baeza-Yates, R. (Eds.). (1992). *Information retrieval: Data structures and algorithms*. Upper Saddle River, NJ, USA: Prentice-Hall Inc.

Fu, A. Y., Wenyin, L., & Deng, X. (2006). Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *IEEE Transactions of Dependable and Secure Computing, 3*(4), 301–311. URL http://dx.doi.org/10.1109/TDSC.2006.50.

Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on recurring malcode. WORM '07* (pp. 1–8). New York, NY, USA: ACM. URL http://dx.doi.org/10.1145/1314389.1314391.

Guan, D., Chen, C., & Lin, J. (2009). Anomaly based malicious URL detection in instant messaging. In *Proceedings of the joint workshop on information security (JWIS)*.

Hansen, R., & Grossman, J. (2008). Clickjacking, URL <http://www.sectheory.com/clickjacking.htm>.

Harley, D., & Bureau, P. -M. (2008). Drive-by downloads from the trenches. In *3rd International Conference on Malicious and Unwanted Software, MALWARE 2008* (pp. 98–103).

Jordan, A. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. *Advances in Neural Information Processing Systems, 14*, 841.

Kan, M., & Thi, H. (2005). Fast webpage classification using URL features. In *Proceedings of the 14th ACM international conference on information and knowledge management* (pp. 325–326). ACM.

Le, A., Markopoulou, A., & Faloutsos, M. (2010). Phishdef: URL names say it all. CoRR abs/1009.2275.

Liu, W., Deng, X., Huang, G., & Fu, A. Y. (2006). An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing, 10*(2), 58–65. URL http://dx.doi.org/10.1109/MIC.2006.23.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110.

Lucca, G. A. D., Fasolino, A. R., Mastoianni, M., & Tramontana, P. (2004). Identifying cross site scripting vulnerabilities in web applications. In *Proceedings of the web site evolution, sixth IEEE international workshop. WSE '04* (pp. 71–80). Washington, DC, USA: IEEE Computer Society. URL <http://dl.acm.org/citation.cfm?id=1025133.1026460>.

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009a). Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '09* (pp. 1245–1254). New York, NY, USA: ACM. URL http://dx.doi.org/10.1145/1557019.1557153.

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009b). Identifying suspicious URLs: An application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning. ICML '09* (pp. 681–688). New York, NY, USA: ACM. URL http://dx.doi.org/10.1145/1553374.1553462.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. L. Cam & J. Neyman (Eds.). *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). University of California Press.

Malware (2011). A great year for malware. Computer fraud and security 2011 (1), 20. URL <http://www.sciencedirect.com/science/article/pii/S1361372311700082>.

Mcgrath, D. K., & Gupta, M. (2008). Behind phishing: An examination of phisher modi operandi. In *Proceedings of the USENIX workshop on large-scale exploits and emergent threats*.

Okanovic, V., & Mateljan, T. (2011). Designing a new web application framework. In *MIPRO, 2011 proceedings of the 34th international convention* (pp. 1315–1318).

Phishtank (2013). Phistank. URL <http://www.phishtank.com>.

Porter, M. F. (1997). Readings in information retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Ch. An algorithm for suffix stripping (pp. 313–316). URL <http://dl.acm.org/citation.cfm?id=275537.275705>.

Provos, N., Mavrommatis, P., Rajab, M. A., & Monrose, F. (2008). All your iFrames point to us. In *Proceedings of the 17th conference on security symposium. SS'08* (pp. 1–15). Berkeley, CA, USA: USENIX Association. URL <http://www.dl.acm.org/citation.cfm?id=1496711.1496712>.

Robertson, S., & Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science, 27*(3), 129–146.

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval. SIGIR '94* (pp. 232–241). New York, NY, USA: Springer-Verlag New York Inc.. URL <http://dl.acm.org/citation.cfm?id=188490.188561>.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for text categorization: Papers from the 1998 workshop* (Vol. 62).

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 24*(5), 513–523. URL http://dx.doi.org/10.1016/0306-4573(88)90021-0.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communication ACM, 18*(11), 613–620. URL http://dx.doi.org/10.1145/361219.361220.

Sood, A. K., & Enbody, R. J. (2011). Frame trapping the frame busting defence. *Network Security, 2011*(10), 8–12. URL <http://www.sciencedirect.com/science/article/pii/S1353485811701052>.

Spertus, E. (1997). Smokey: Automatic recognition of hostile messages. In *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on innovative applications of artificial intelligence. AAAI'97/IAAI'97* (pp. 1058–1065). AAAI Press. URL <http://dl.acm.org/citation.cfm?id=1867406.1867616>.

Yang, Y., & Chute, C. G. (1994). An example-based mapping method for text categorization and retrieval. *ACM Transactions of Information Systems, 12*(3), 252–277. URL http://dx.doi.org/10.1145/183422.183424.