9

Hardening Email Security via Bayesian Additive Regression Trees

Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang and Suku Nair
SMU HACNet Lab, Southern Methodist University Dallas, TX, USA

1. Introduction

The changeable structures and variability of email attacks render current email filtering solutions useless. Consequently, the need for new techniques to harden the protection of users' security and privacy becomes a necessity. The variety of email attacks, namely *spam*, damages networks' infrastructure and exposes users to new attack vectors daily. Spam is unsolicited email which targets users with different types of commercial messages or advertisements. Porn-related content that contains explicit material or commercials of exploited children is a major trend in these messages as well. The waste of network bandwidth due to the numerous number of spam messages sent and the requirement of complex hardware, software, network resources, and human power are other problems associated with these attacks. Recently, security researchers have noticed an increase in malicious content delivered by these messages, which arises security concerns due to their attack potential. More seriously, *phishing* attacks have been on the rise for the past couple of years. Phishing is the act of sending a forged e-mail to a recipient, falsely mimicking a legitimate establishment in an attempt to scam the recipient into divulging private information such as credit card numbers or bank account passwords (James, 2005). Recently phishing attacks have become a major concern to financial institutions and law enforcement due to the heavy monetary losses involved. According to a survey by Gartner group, in 2006 approximately 3.25 million victims were spoofed by phishing attacks and in 2007 the number increased by almost 1.3 million victims. Furthermore, in 2007, monetary losses, related to phishing attacks, were estimated by $3.2 billion. All the aforementioned concerns raise the need for new detection mechanisms to subvert email attacks in their various forms.
Despite the abundance of applications available for phishing detection, unlike spam classification, there are only few studies that compare machine learning techniques in predicting phishing emails (Abu-Nimeh et al., 2007). We describe a new version of Bayesian Additive Regression Trees (BART) and apply it to phishing detection. A phishing dataset is constructed from 1409 raw phishing emails and 5152 legitimate emails, where 71 features (variables) are used in classifiers' training and testing. The variables consist of both textual and structural features that are extracted from raw emails. The performance of six classifiers, on this dataset, is compared using the area under the curve (AUC) (Huang & Ling, 2005). The classifiers include Logistic Regression (LR), Classification and Regression Trees (CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random

Source: Machine Learning, Book edited by: Abdelhamid Mellouk and Abdennacer Chebira,
ISBN 978-3-902613-56-1, pp. 450, February 2009, I-Tech, Vienna, Austria

Forests (RF), and Neural Networks (NNet). In addition to the AUC, additional measures are used to gauge the classifiers' performance, such as the error rate, false positive, and false negative rates.

1.1 Motivation to Bayesian methodology

We start by providing a discussion on Bayesian learning and the reasons behind choosing BART among another classifiers, then we illustrate the technical details of BART. BART is a Bayesian approach, thus it inherits all the advantages from Bayesian learning. There are various advantages of Bayesian learning when compared to other statistical approaches. As opposed to the frequentist approach for defining the probability of an uncertain event, here one needs a record of past information of an event. Yet, if this information is not available, then the frequentist approach cannot be used to define the degree of belief in the uncertain event. On the other hand, the Bayesian approach allows us to reason about beliefs under conditions of uncertainty. Thus, it helps in modeling uncertainty in a probabilistic way (Neal, 1995). In addition, Bayesian inference is regarded as a good approach to tackle the problem of data modeling (Kandola, 2001). A model is designed for a particular application and adapted according to the data while the data arrives from the application. The model then provides a representation of the prior beliefs about the application and the information derived from the data (Bishop, 1995).

Another advantage of the Bayesian approach is that one does not need to update the model entirely when acquiring new knowledge. Yet, the new data can be used to update the current model instead of re-fitting the entire model. This feature comes handy especially in phishing detection since phishing attacks change frequently and vastly to lure filters and detection mechanisms. Assuming that one needs to re-fit the entire model when new batch of emails arrives, the procedure becomes very computationally extensive and time consuming, thus impractical.

Furthermore, BART is a model-based Bayesian approach. As opposed to to those algorithm-based learning methods, model-based approaches can provide full and accurate assessment of uncertainty in predictions, while remaining highly competitive in terms of predictive accuracy. In addition, model-based approaches are considered non-greedy, hence opposed to selecting the best solution at the time being and not worrying about the future (i.e. whether the solution is efficient or not), the solution is interchangeable accordingly. Also, model-based approaches are non-adhoc, hence the provided solution is not only selected to a particular problem; however, it can be used as a general case.

1.2 Why Bayesian additive regression trees?

BART automatically selects variables from a large pool of input predictors, while searching for models with highest posterior probabilities for future prediction, via a backfitting Markov chain Monte Carlo (MCMC) algorithm (see section 3.1 for further details). Compared to other Bayesian methods, such as Naive Bayes and Bayesian Networks, the latter approaches require variable selection to be done separately, otherwise they use all the variables supplied for training, thus the performance of the classifier will be very poor. Also, it is well known that variable selection in a high dimensional space is a very difficult problem that often requires intensive computations. As we mentioned earlier, phishing emails change regularly and vastly to lure detection mechanisms and the variables may change over time as well. Yet, the above nice feature of BART comes handy when training

on newly arriving emails on a regular basis. With no additional requirements to perform variable selection, BART simultaneously accomplishes variable selection during the training phase.

In addition, in phishing detection hundreds of potential features are extracted from raw emails. Only an unknown subset of them are useful for prediction and others are irrelevant. Blindly including all the variables in the step of training often leads to overfitting, and hence predicting new attacks may be poor. However, with the automatic variable selection feature in BART this problem is solved.

Further, many other Bayesian learning approaches require very careful prior specification, and hence extra effort and operational cost in training models. However, BART, as shown in (Chipman et al., 2006), appears to be relatively insensitive to small changes in the prior specification and the choice of the number of trees. According to (Chipman et al., 2006), the default priors work very well, which enables BART to be an objective and automatic training procedure. This is desirable in situations in which there is no prior information available or no human intervention is preferred.

Furthermore, BART is a class of Bayesian additive models with multivariate components of binary trees. Using binary trees as model components makes BART more exible in practice, as opposed to common regression approaches, since the structure of binary trees has been proved to approximate well nonlinear and nonsmooth functional forms in many applications (Hastie et al., 2001).

Also, BART uses a sum-of-trees-model which is more exible than any single tree model that can hardly account for additive effects. Each tree component is regarded as a *weak learner*, which explains a small and different part of the unknown relationship between the input and output. In addition, multivariate components of BART can easily incorporate high-order interaction effects among three or more input variables, which can be dificult to capture by other additive models.

Moreover, the rich structure of BART leads to its excellent learning ability, even in the presence of a very complicated structure embedded in data. Since phishing emails look very similar to legitimate emails, actually they are duplicates of legitimate emails with some changes, learning is a challenging problem and perhaps involves discovering an elaborate and subtle relationship from data.

2. Related work

(Chandrasekaran et al., 2006) proposed a technique to classify phishing based on structural properties of phishing emails. They used a total of 25 features mixed between style markers (e.g. the words suspended, account, and security) and structural attributes, such as the structure of the subject line of the email and the structure of the greeting in the body. They tested 200 emails (100 phishing and 100 legitimate). They applied simulated annealing as an algorithm for feature selection. After a feature set was chosen, they used information gain (IG) to rank these features based on their relevance. They applied one-class SVM to classify phishing emails based on the selected features. Their results claim a detection rate of 95% of phishing emails with a low false positive rate.

(Fette et al., 2007) compared a number of commonly-used learning methods through their performance in phishing detection on a past phishing data set, and finally Random Forests were implemented in their algorithm PILFER. Their methods can be used to detect phishing websites as well. They tested 860 phishing emails and 6950 legitimate emails. The proposed

method detected correctly 96% of the phishing emails with a false positive rate of 0.1%. They used ten features handpicked for training and their phishing dataset was collected in 2002 and 2003. As pointed out by the authors themselves, their implementation is not optimal and further work in this area is warranted.
(Abu-Nimeh et al., 2007) compared six machine learning techniques to classify phishing emails. Their phishing corpus consisted of a total of 2889 emails and they used 43 features (variables). They showed that, by merely using a *bag-of-words* approach, the studied classifiers could successfully predict more than 92% of the phishing emails. In addition, the study showed that Random Forests achieved the maximum predictive accuracy and Logistic Regression achieved the minimum false positives on the studied corpus.

3. Machine learning approaches for binary classification

In the literature, there exist several machine learning techniques for binary classification, e.g., logistic regression, neural networks (NNet), binary trees and their derivatives, discriminant analysis (DA), Bayesian networks (BN), nearest neighbor (NN), support vector machines (SVM), boosting, bagging, etc. The interested reader can refer to (Hastie et al., 2001) and the references therein for a detailed overview. Here we describe the application of Bayesian Additive Regression Trees (BART) for learning from data, combined with a probit setup for binary responses, to detect phishing emails.
Most of the machine learning algorithms discussed here are categorized as *supervised* machine learning, where an algorithm (classifier) is used to map inputs to desired outputs using a specific function. In classification problems a classifier tries to learn several features (variables or inputs) to predict an output (response). Specifically in phishing classification, a classifier will try to classify an email to phishing or legitimate (response) by learning certain characteristics (features) in the email.
Applying any supervised machine learning algorithm to phishing detection consists of two steps: training and classification. During the *training* step a set of compiled phishing and non-phishing messages (with known status) is provided as training dataset to the classifier. Emails are first transformed into a representation that is understood by the algorithms. Specifically, raw emails are converted to vectors using the vector space model (VSM) (Salton & McGill, 1983), where the vector represents a set of features that each phishing and non-phishing email carries. Then the learning algorithm is run over the training data to create a classifier. The *classification* step follows the training (learning) phase. During classification, the classifier is applied to the vector representation of real data (i.e. test dataset) to produce a prediction, based on learned experience.

3.1 Bayesian additive regression trees

Bayesian Additive Regression Trees (BART) is a new learning technique, proposed by (Chipman et al., 2006), to discover the unknown relationship between a continuous output and a dimensional vector of inputs. The original model of BART was not designed for classification problems, therefore, a modified version, hereafter CBART, which is applicable to classification problems in general and phishing classification in particular is used. Note that BART is a learner to predict quantitative outcomes from observations via regression. There is a distinction between regression and classification problems. Regression is the process of predicting quantitative outputs. However, when predicting qualitative

(categorical) outputs this is called a classification problem. Phishing prediction is a binary classification problem, since we measure two outputs of email either phishing =1 or legitimate =0 (Hastie et al., 2001).

BART discovers the unknown relationship f between a continuous output Y and a p dimensional vector of inputs $x = (x_1, ..., x_p)$. Assume $Y = f(x) + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$ is the random error. Motivated by ensemble methods in general, and boosting algorithms in particular, the basic idea of BART is to model or at least approximate $f(x)$ by a sum of regression trees,

$$f(x) = \sum_{i=1}^{m} g_i(x); \tag{1}$$

each g_i denotes a binary tree with arbitrary structure, and contributes a small amount to the overall model as a *weak learner*, when m is chosen large. An example of a binary tree structure is given in Figure 1, in which a is the root node, c is an internal node, and b, d and e are three terminal nodes that are associated with parameter μ_1, μ_2 and μ_3, respectively. Also, each of the interior (i.e., non-terminal) nodes is associated with a binary splitting rule based on some x variable. By moving downwards from the root, an observation with given x will be assigned to a unique terminal node, according to the splitting rules associated with the nodes included in its path. In consequence, the corresponding parameter of the terminal node will be the value of g for this observation.

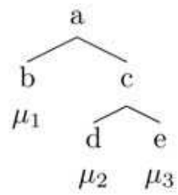


Fig. 1. A binary tree structure

Let T_i be the i^{th} binary tree in the model (1), consisting of a set of decision rules (associated with its interior nodes) and a set of terminal nodes, for $i = 1, ..., m$. Let M_i be the vector containing all terminal node parameters of T_i such that $M = \{M_1, ..., M_{b_i}\}$ and b_i is the number of terminal nodes that T_i has. Now we can explicitly write

$$Y = g(x; T_1, M_1) + \ldots + g(x; T_m, M_m) + \epsilon. \tag{2}$$

Figure 2 depicts an example of a binary tree in the BART model. Note that the BART contains multiple binary trees, since it is an additive model. Each node in the tree represents a feature in the dataset and the terminal nodes represent the probability that a specific email is phishing, given that it contains certain features. For example, if an email contains HTML code, contains javascript, and the javascript contains form validation, then the probability that this email is phishing is 80% (refer to Figure 2). These features are discussed in more details in Section 4.1.1.

BART is fully model-based and Bayesian in the sense that a *prior* is specified, a *likelihood* is defined using the data, and then a sequence of draws from the *posterior* using Markov chain Monte Carlo (MCMC) is obtained. Specifically, a *prior* distribution is needed for T, M, and σ, respectively. Each draw represents a fitted model f^* of the form (1).

To specify a *prior* distribution $P(T)$ on T, one needs three pieces of information; (i) determining how likely a node will be split when a tree is created; (ii) determining which variable will be chosen to split the node; (iii) determining the rule that will be used for splitting. The main goal here is to generate small trees or "*weak learners*", hence each tree plays a small share in the overall fit, but when combined all produce a powerful "committee".

For the *prior* distribution on terminal node parameters $P(\mu)$, the parameters of the terminal nodes are assumed independent *a priori*, hence the prior mean $E(Y|x) = \sum_{i=1}^{m} \mu_i$. Lastly, for the variance of noise σ^2, a prior $P(\sigma)$ is needed. The parameters of the prior on σ can be specified from a *least square linear regression* of Y on the original x's.

Now given the *prior* distributions a *backfitting MCMC Gibbs sampler* is used to sample from the *posterior* distribution as shown below.

Repeat $i = 1$ to I (say $I = 1000$, where I is the number of simulations):

- Sample T_j conditional on Y, all Ts but T_j, all μs, and σ.
- Sample M_j given all Ts, all Ms but M_j, and σ.
- Repeat the above steps m times for $j = 1, ., m$, where j is the total number of trees available.
- Sample σ given Y and all Ts, all Ms and σ.

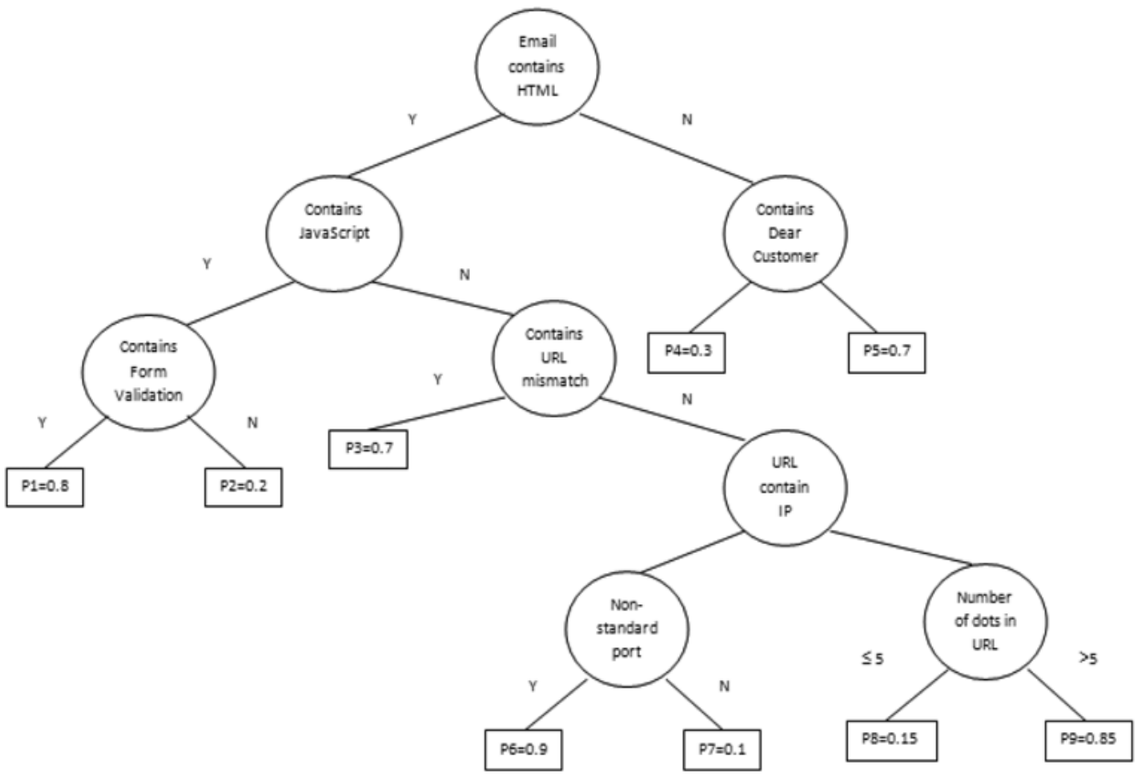


Fig. 2. Example of a binary tree.

Since this is a Markov chain, simulation i depends on simulation i - 1. The MCMC simulation changes tree structures based on a stochastic tree generating process. The structures can be changed by randomly using any of the following four actions. *Grow* can be applied to grow a new pair of terminal nodes from a terminal node and make it become an

interior one. *Prune* can be applied to prune a pair of terminal nodes and make their parent node become a terminal one. *Change* is to change a splitting rule of a non-terminal node. *Swap* is to swap rules between a parent node and a child. By these changes, MCMC generates different tree structures and chooses the tree structure that provides the "best" sum-of-trees model according to posterior probabilities of trees.

It is worth mentioning that BART has several appealing features, which make it competitive compared to other learning methods and motivate our study as well. Rather than using a single regression tree, BART uses a sum- of-trees model that can account for additive effects. Also, the binary tree structure helps in approximating well nonlinear and non-smooth relationships (Hastie et al., 2001). Furthermore, BART can conduct automatic variable selection of inputs while searching for models with highest posterior probabilities during MCMC simulation. In addition, by applying Bayesian learning, BART can use newly coming data to update the current model instead of re-fitting the entire model.

Despite the advantages mentioned earlier, it is well known that a Bayesian approach usually brings heavy computation time due to its nature. Predicting the *posterior* probabilities via MCMC is usually time consuming and requires complex computations.

3.1.1 BART for classification (CBART)

As mentioned in Section 3.1, BART requires the output variable to be continuous, instead of binary. Let $Y = 1$ if an email is phishing; otherwise $Y = 0$. To use BART with binary outputs, we introduce a latent variable Z in connection with Y in spirit of (Albert & Chib, 1993), by defining

$$\begin{aligned} Z &= f(x) + \epsilon, \quad \epsilon \sim N(0,1); \\ Y &= \begin{cases} 1 & \text{if } Z > 0; \\ 0 & \text{if } Z \leq 0. \end{cases} \end{aligned} \tag{3}$$

where $f(x)$ is the sum-of-trees model in (1). Note here, we fix σ at 1, due to the simple binary nature of Y . This yields the probit link function between the phishing probability p and $f(x)$,

$$p \equiv P(Y = 1|x) = P(Z > 0|x) = \Phi(f(x)), \tag{4}$$

where $\Phi(\cdot)$ is the cumulative density function of $N(0, 1)$.

Under the above setup of the latent variable Z, we can use BART to learn $f(x)$ from data, after appropriately modifying the prior distribution on M and the MCMC algorithm proposed in (Chipman et al., 2006) for posterior computation. Then we can estimate $Y = 1$ if the fitted $f^*(x) > 0$, otherwise estimate $Y = 0$. Further, we can obtain the estimate of p through equation (4).

Before we describe the algorithm, let T denote a binary tree consisting of a set of interior node decision rules and a set of terminal nodes, and let $M = \{\mu_1, \mu_2, ..., \mu_b\}$ denote a set of parameter values associated with each of the b terminal nodes of T. Now we explicitly denote the ith component of the model $g_i(x)$ by $g_i(x; T_i, M_i)$. Also, let $T_{(j)}$ be the set of all trees in the sum (1) except T_j , and $M_{(j)}$ the associated terminal node parameters. Let y denote the observed phishing status of emails in the training data. The algorithm will generate draws from the posterior distribution

$$p((T_1, M_1), ..., (T_m, M_m), Z|y) \tag{5}$$

rather than drawing from

$$p((T_1, M_1), ..., (T_m, M_m), \sigma|y)$$

in the original algorithm. A typical draw from the new posterior (5) entails m successive draws of tree component(T_j ,M_j) conditionally on ($T_{(j)}$,$M_{(j)}$,Z):

$$\begin{array}{c} (T_1, M_1)|T_{(1)}, M_{(1)}, y, Z \\ (T_2, M_2)|T_{(2)}, M_{(2)}, y, Z \\ \vdots \\ (T_m, M_m)|T_{(m)}, M_{(m)}, y, Z \end{array} \tag{6}$$

followed by a draw of Z from the full conditional:

$$Z|(T_1, M_1), ..., (T_m, M_m), y. \tag{7}$$

Note that there is no need to draw σ in our new algorithm since it is set to 1.

We proceed to discuss how to implement (6) and (7). First, we claim that the first step is essentially the same as in the original algorithm. This is because in (6), no extra information is given by y when Z is given, since y can be completely determined by Z through (3). Hence we can remove the redundant y in (6), and use the original algorithm (substitute y by Z and set σ = 1) to draw from (6). Since Z is latent, we need an extra step to draw values of Z from (7). It can be verified that for the jth email in the training data, Z_j | (T_1,M_1), ..., (T_m,M_m), y is distributed as $N(\sum_{i=1}^{m} g_i(x; T_i,M_i), 1)$ truncated at the left by 0 if y_j = 1, and distributed as $N(\sum_{i=1}^{m} g_i(x; T_i,M_i), 1)$ truncated at the right by 0 if y_j = 0. Thus, drawing from (7) can be easily done by drawing values from the normal distributions and then truncating them by 0 either from the right or from the left based on the value of y.

As shown above, BART is well suited for binary classification, under the probit setup with the use of the latent variable. In this case, it is even easier than before because σ is no longer an unknown parameter and the draws of Z are extremely easy to obtain.

We now briey discuss how to use BART for prediction. In an MCMC run, we can simply pick up the "best" f* (according to posterior probabilities or Bayes factor or other criteria) from the sequence of visited models, and save it for future prediction. Note that the selected f* perhaps involves a much less number of input variables than p since BART automatically screens input variables. This would allow prediction for a new email to be quickly done since much less information needs to be extracted from the email. A better way is to use the posterior mean of f for prediction, approximated by averaging the f * over the multiple draws from (5), and further gauge the uncertainty of our prediction by the variation across the draws. However, this involves saving multiple models in a physical place for future use. A more realistic approach is to use the best B fitted models for prediction that account for the 95% posterior probabilities over the space of sum-of-tree models. Usually, B is a number less than 20 and again, when predicting a new email is or not, a much less number of input variables than p are expected to be used (Abu-Nimeh et al., 2008).

3.2 Classification and regression trees

CART or Classification and Regression Trees (Breiman et al., 1984) is a model that describes the conditional distribution of y given x. The model consists of two components; a tree T

with b terminal nodes, and a parameter vector $\Theta = (\theta_1, \theta_2, \ldots, \theta_b)$ where θ_i is associated with the i^{th} terminal node. The model can be considered a classification tree if the response y is discrete or a regression tree if y is continuous. A binary tree is used to partition the predictor space recursively into distinct homogenous regions, where the terminal nodes of the tree correspond to the distinct regions. The binary tree structure can approximate well non-standard relationships (e.g. non-linear and non-smooth). In addition, the partition is determined by splitting rules associated with the internal nodes of the binary tree. Should the splitting variable be continuous, a splitting rule in the form $\{x_i \in C\}$ and $\{x_i \notin C\}$ is assigned to the left and the right of the split node respectively. However, should the splitting variable be discrete, a splitting rule in the form $\{x_i \leq s\}$ and $\{x_i > s\}$ is assigned to the right and the left of the splitting node respectively (Chipman et al., 1998).
CART is exible in practice in the sense that it can easily model nonlinear or nonsmooth relationships. It has the ability of interpreting interactions among predictors. It also has great interpretability due to its binary structure. However, CART has several drawbacks such as it tends to overfit the data. In addition, since one big tree is grown, it is hard to account for additive effects.

3.3 Logistic regression

Logistic regression is the most widely used statistical model in many fields for binary data (0/1 response) prediction, due to its simplicity and great interpretability. As a member of generalized linear models it typically uses the *logit* function. That is

$$log \frac{P(x;\beta)}{1 - P(x;\beta)} = \beta^T x$$

where x is a vector of p predictors $x = (x_1, x_2, \ldots, x_p)$, y is the binary response variable, and β is a $p \times 1$ vector of regression parameters.
Logistic regression performs well when the relationship in the data is approximately linear. However, it performs poorly if complex nonlinear relationships exist between the variables. In addition, it requires more statistical assumptions before being applied than other techniques. Also, the prediction rate gets affected if there is missing data in the data set.

3.4 Neural networks

A neural network is structured as a set of interconnected identical units (neurons). The interconnections are used to send signals from one neuron to the other. In addition, the interconnections have weights to enhance the delivery among neurons (Marques de Sa, 2001). The neurons are not powerful by themselves, however, when connected to others they can perform complex computations. Weights on the interconnections are updated when the network is trained, hence significant interconnection play more role during the testing phase. Figure 3 depicts an example of neural network. The neural network in the figure consists of one input layer, one hidden layer, and one output layer. Since interconnections do not loop back or skip other neurons, the network is called *feedforward*. The power of neural networks comes from the nonlinearity of the hidden neurons. In consequence, it is signi_cant to introduce nonlinearity in the network to be able to learn complex mappings. The commonly used function in neural network research is the *sigmoid* function, which has the form (Massey et al., 2003)

$$a(x) = \frac{1}{1 + e^{-x}}$$

Although competitive in learning ability, the fitting of neural network models requires some experience, since multiple local minima are standard and delicate regularization is required.

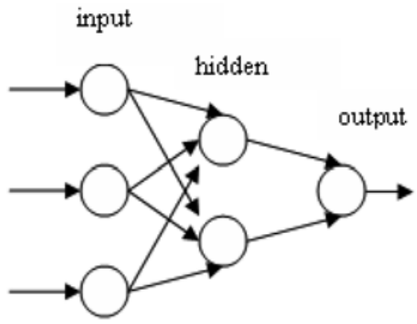


Fig. 3. Neural Network.

3.5 Random forests

Random forests are classi_ers that combine many tree predictors, where each tree depends on the values of a random vector sampled independently. Furthermore, all trees in the forest have the same distribution (Breiman, 2001). In order to construct a tree we assume that n is the number of training observations and p is the number of variables (features) in a training set. In order to determine the decision node at a tree we choose $k << p$ as the number of variables to be selected. We select a *bootstrap* sample from the n observations in the training set and use the rest of the observations to estimate the error of the tree in the testing phase. Thus, we randomly choose k variables as a decision at a certain node in the tree and calculate the best split based on the k variables in the training set. Trees are always grown and never pruned compared to other tree algorithms.

Random forests can handle large numbers of variables in a data set. Also, during the forest building process they generate an internal unbiased estimate of the generalization error. In addition, they can estimate missing data well. A major drawback of random forests is the lack of reproducibility, as the process of building the forest is random. Further, interpreting the final model and subsequent results is difficult, as it contains many independent decisions trees.

3.6 Support vector machines

Support Vector Machines (SVM) are one of the most popular classifiers these days. The idea here is to find the optimal separating hyperplane between two classes by maximizing the margin between the classes closest points. Assume that we have a linear discriminating function and two linearly separable classes with target values +1 and -1. A discriminating hyperplane will satisfy:

$$w^{'} x_i + w_0 \geq 0 \text{ if } t_i = +1;$$

$$w^{'} x_i + w_0 < 0 \text{ if } t_i = -1$$

Now the distance of any point x to a hyperplane is $\mid w'x_i+w_0 \mid / \parallel w \parallel$ and the distance to the origin is $\mid w_0 \mid / \parallel w \parallel$. As shown in Figure 4 the points lying on the boundaries are

called support vectors, and the middle of the margin is the optimal separating hyperplane that maximizes the margin of separation (Marques de Sa, 2001).
Though SVMs are very powerful and commonly used in classification, they suffer from several drawbacks. They require high computations to train the data. Also, they are sensitive to noisy data and hence prone to overfitting.

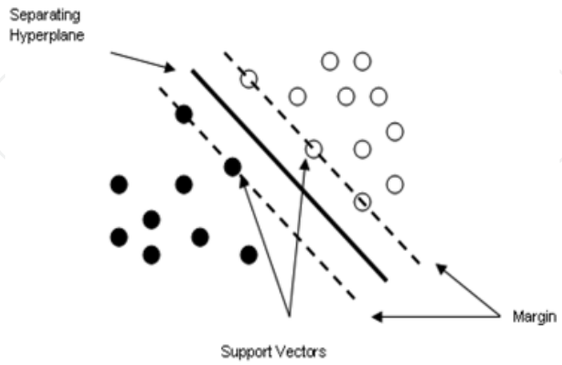


Fig. 4. Support Vector Machines.

4. Quantitative evaluation

4.1 Phishing dataset

The phishing dataset constitutes of 6561 raw emails. The total number of phishing emails in the dataset is 1409 emails. These emails are donated by (Nazario, 2007) covering many of the new trends in phishing and collected between August 7, 2006 and August 7, 2007. The total number of legitimate email is 5152 emails. These emails are a combination of financial-related and other regular communication emails. The financial-related emails are received from financial institutions such as Bank of America, eBay, PayPal, American Express, Chase, Amazon, AT&T, and many others. As shown in Table 1, the percentage of these emails is 3% of the complete dataset. The other part of the legitimate set is collected from the authors' mailboxes. These emails represent regular communications, emails about conferences and academic events, and emails from several mailing lists.

Corpus	No. of Emails	Percentage (%)
Phishing	1409	21%
Legitimate (financial)	178	3%
Legitimate (other)	4974	76%
Total	6561	100%

Table 1. Corpus description.

4.1.1 Data standardization, cleansing, and transformation

The analysis of emails consists of two steps: First, *textual analysis*, where text mining is performed on all emails. In order to get consistent results from the analysis, one needs to standardize the studied data. Therefore, we convert all emails into XML documents after stripping all HTML tags and email header information. Figure 5 shows an example of a phishing email after the conversions. Text mining is performed using the text-miner software kit (TMSK) provided by (Weiss et al., 2004). Second, *structural analysis*. In this step

we analyze the structure of emails. Specifically, we analyze links, images, forms, javascript code and other components in the emails.

```
<DOC>
 <BODY>
   Dear eBay User ,
   After fraud complaints from the eBay
   members, the eBay Inc. had developed a security program against the
   fraudulend attempts of accounts thefts. For that we have to securise
   all the members informations by updating and checking the
   registrated informations. Please update  your information by
   completing the form from the forwarded link so we can check your
   account validity and your identity and login to eBay in order to
   update your informations. This process will take 5 days, period when
   you will not be able to acces your eBay account. After this period
   you will receive instructions to enter and securise your eBay
   account.Please click the link below and sign in into your
   account:
     http://signin.ebay.com/aw-cgi/eBayISAPI.dll?
     SignIn&amp;ssPageName=h:h:sin:US
   As outlined in our User Agreement, eBay will periodically send you
   information about site changes and enhancements. Visit our Privacy
   Policy and User Agreement if you have any questions.
   Regards,Safeharbor Department eBay, Inc.
 </BODY>
 <TOPICS><TOPIC>phish<TOPIC><TOPICS>
</DOC>
```

Fig. 5. Phishing email after conversion to XML.

Afterwards, each email is converted into a vector $\vec{x} = \langle x_1, x_2, ..., x_p \rangle$, where $x_1, ..., x_p$ are the values corre- sponding to a specific feature we are interested in studying (Salton & McGill, 1983). Our dataset consists of 70 continuous and binary features (variables) and one binary response variable, which indicates that email is phishing=1 or legitimate=0. The first 60 features represent the frequency of the most frequent terms that appear in phishing emails. Choosing words (terms) as features is widely applied in the text mining literature and is referred to as "bag-of-words". In Table 2 we list both textual and structural features used in the dataset. As shown in Figure 6, we start by striping all *attachments* from emails in order to facilitate the analysis of emails. The following subsections illustrate the textual and structural analysis in further details.

4.1.2 Textual analysis

As we mentioned earlier we start by stripping all *attachments* from email messages. Then, we extract the *header* information of all emails keeping the email body. Afterwards, we extract the *html tags* and *elements* from the body of the emails, leaving out the body as plain text. Now, we standardize all emails in a form of XML documents. The `<DOC> </DOC>` tags indicate the beginning and ending of a document respectively. The `<BODY> </BODY>` tags indicate the starting and ending of an email body respectively. The `<TOPICS> </TOPICS>` tags indicate the class of the email, whether it is phish or legit (see Figure 5).

Feature	Binary/Continuous	Value	Description
1	binary	0/1	class of email, phishing or legitimate
2-61	continuous	TF/IDF	frequency of terms
62	binary	0/1	link mismatch
63	binary	0/1	URL contains IP
64	binary	0/1	email contains javascript
65	binary	0/1	email contains images that link to external server
66	binary	0/1	email contains form validation
67	binary	0/1	URL contains non-standard ports
68	continuous	maximum total	total number of dots in URL
69	continuous	total	total number of links in email
70	binary	0/1	email contains URL redirection
71	binary	0/1	email contains URL encoding

Table 2. Feature description.

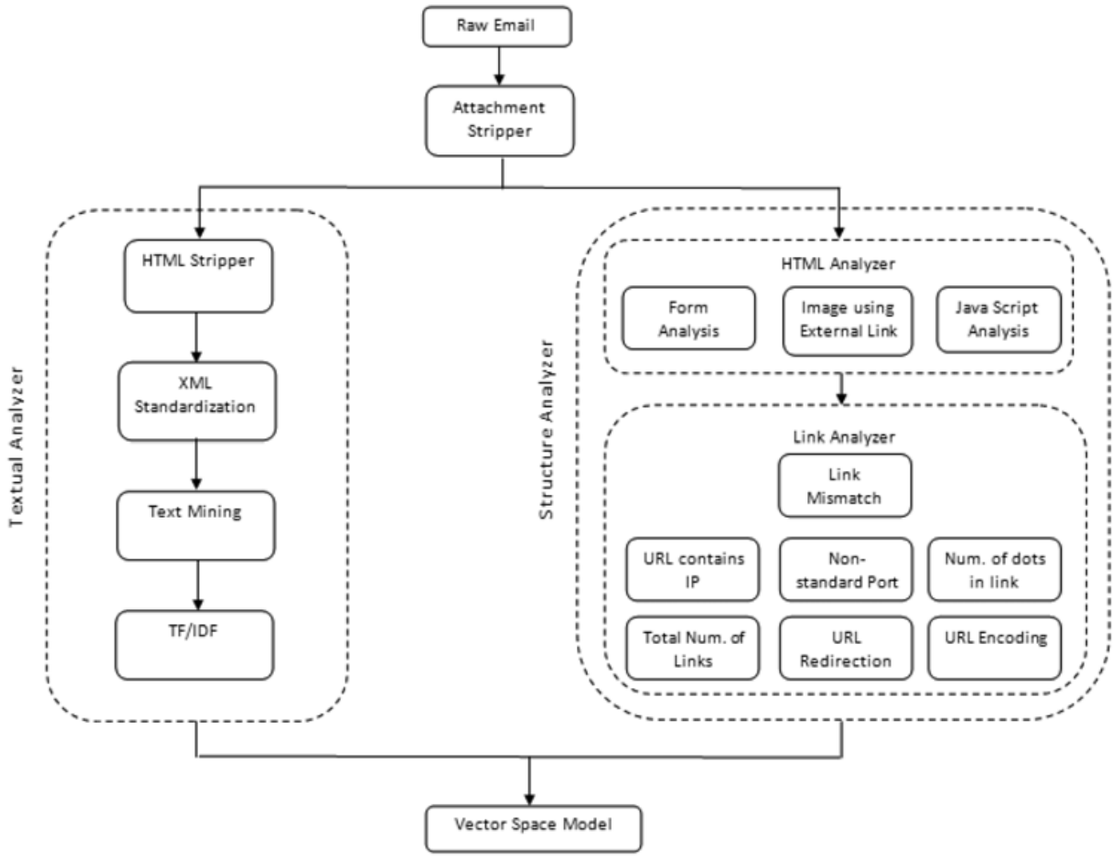


Fig. 6. Building the phishing dataset.

Thus, we filter out *stopwords* from the text of the body. We use a list of 816 commonly used English stopwords. Lastly, we find the most frequent terms using TF/IDF (Term Frequency Inverse Document Frequency) and choose the top 60 most frequent terms that appear in

phishing emails. TF/IDF calculates the number of times a word appears in a document multiplied by a (monotone) function of the inverse of the number of documents in which the word appears. In consequence, terms that appear often in a document and do not appear in many documents have a higher weight (Berry, 2004).

4.1.3 Structural analysis

Textual analysis generates the first 60 features in the dataset and the last 10 features are generated using structural analysis. Unlike textual analysis, here we only strip the attachments of emails keeping HTML tags and elements for further analysis. First, we perform HTML analysis, in which we analyze *form* tags, *javascript* tags, and *image* tags. Legitimate emails rarely contain *form* tags that validate the user input. Phishing emails, on the other hand, use this techniques to validate victims' credentials before submitting them to the phishing site. In consequence, if an email contains a *form* tag, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 7 shows an example of a Federal Credit Union phish which contains a *form* tag.

```
<FORM id=Form1 name=CreditCard action=/Credit/CC_Input.asp
  method=post>
   The National Credit Union Administration (NCUA) is committed
   to maintain ... Thank you for using Federal Credit Union.
</FORM>
```

Fig. 7. Form validation in phishing email.

Similarly, legitimate emails rarely contain *javascript*, however, phishers use javascript to validate users input or display certain elements depending on the user input. If the email contains javascript, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 8 shows an example of javascript that is used by a phisher to validate the victims account number.

```
<SCRIPT language=javascript>
 function checkSendEmailForm(){
     if (document.emailSubmission.accountNum.value==""){
       alert('Please enter your account number
              before you push the send button');
     }
     else{
      document.emailSubmission.submit();
     }
 }
</SCRIPT>
```

Fig. 8. Javascript to validate account number.

Spammers have used images that link to external servers in their emails, also dubbed as *Web beacons*, to verify active victims who preview or open spam emails. Phishers also have been following the same technique to verify active victims and also to link to pictures from legitimate sites. We analyze emails that contain *image* tags that link to external servers. If the

email contains such an image, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 9 shows an example of a image tag with an external link.

```
<a href="http://201.128.53.64/index.php">
 <img src="http://pics.ebaystatic.com/aw/pics/navbar/eBayLogoTM.gif">
</a>
```

Fig. 9. Image linking to an external server.

The second part in structural analysis involves the *link analysis* process. Here we analyze links in emails. It is well known that phishers use several techniques to spoof links in emails and in webpages as well to trick users into clicking on these links. When analyzing links we look for link mismatch, URL contains IP address, URL uses non-standard ports, the maximum total number of dots in a link, total number of links in an email, URL redirection, and URL encoding. In what follows we describe these steps in more details.
When identifying a link mismatch we compare links that are displayed to the user with their actual destination address in the `<a href>` tag. If there is a mismatch between the displayed link and the actual destination in any link in the email, then the corresponding feature in the dataset is set to 1, otherwise, it is set to 0. Figure 10 shows an example of a PayPal phish, in which the phisher displays a legitimate Paypal URL to the victim; however, the actual link redirects to a Paypal phish.

```
<a href="http://218.214.124.67/paypal/cgi-bin/index.php">
 https://www.paypal.com/cgi-bin/webscr?cmd=_login-run
</a>
```

Fig. 10. URL mismatch in link.

A commonly used technique, but easily detected even by naive users, is the use of IP addresses in URLs (i.e. unresolved domain names). This has been and is still seen in many phishing emails. It is unlikely to see unresolved domain names in legitimate emails; however, phishers use this technique frequently, as it is more convenient and easier to setup a phishing site. If the email contains a URL with an unresolved name, then the corresponding feature in the dataset is set to 1, otherwise, it is set to 0. Phishers often trick victims by displaying a legitimate URL and hiding the unresolved address of the phishing site in the `<a href>` tag as shown in the example in Figure 10.
Since phishing sites are sometimes hosted at compromised sites or botnets, they use non-standard port numbers in URLs to redirect the victim's traffic. For example instead of using port 80 for http or port 443 for https traffic, they use different port numbers. If the email contains a URL that redirects to a non-standard port number, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 11 shows an example of a phishing URL using a non-standard port number.

```
To receive email notifications in plain text instead of HTML, update your preferences, <a
href="http://www.online-paypal.h1x.com:8088/security.htm"> Click here.  </a>
```

Fig. 11. Phishing URL using non-standard port number.

We count the number of links in an email. Usually, phishing emails contain more links compared to legitimate ones. This is a commonly used technique in spam detection, where messages that contain a number of links more than a certain threshold are filtered as spam.

However, since phishing emails are usually duplicate copies of legitimate ones, this feature might not help in distinguishing phishing from financial-related legitimate emails; however, it helps in distinguishing phishing from other regular legitimate messages.
Since phishing URLs usually contain multiple sub-domains so the URL looks legitimate, the number of dots separating sub-domains, domains, and TLDs in the URLs are usually more than those in legitimate URLs. Therefore, in each email we find the link that has the maximum number of dots. The maximum total number of dots in a link in an email thus is used as a feature in the dataset. Figure 12 shows an example of a Nationwide spoof link. Note the dots separating different domains and sub-domains.

```
http://kaboom-uf.com/vwar/backup/nationwide.co.uk.online.banking.update.compulsory.secure.signon
```

Fig. 12. Number of dots in a Nationwide spoof URL.

Phishers usually use open redirectors to trick victims when they see legitimate site names in the URL. Specifically, they target open redirectors in well known sites such as `aol.com`, `yahoo.com`, and `google.com`. This technique comes handy when combined with other techniques, especially URL encoding, as naive users will not be able to translate the encoding in the URL. Figure 13 shows an example of an AOL open redirector.

```
http://aol.com/redir.adp?_url=http://64-60-13-140.static-ip.telepacific.net:82/ebay.com/reg.php
```

Fig. 13. Open redirector at AOL.

The last technique that we analyze here is URL encoding. URL encoding is used to transfer characters that have a special meaning in HTML during http requests. The basic idea is to replace the character with the "%" symbol, followed by the two-digit hexadecimal representation of the ISO-Latin code for the character. Phishers have been using this approach to mask spoofed URL and hide the phony addresses of these sites. However, they encode not only special characters in the URL, but also the complete URL. As we mentioned earlier, when this approach is combined with other techniques, it makes the probability of success for the attack higher, as the spoofed URL looks more legitimate to the naive user. Figure 14 shows an example of URL encoding combined with URL redirection.

```
http://www.aol.com/redir.adp?_url=%31%30%30%26%41%64%49%44%3D%34%34%39
```

Fig. 14. URL encoding combined with URL redirection.

Figure 6 depicts a block diagram of the approach used in building the dataset. It shows both textual and structural analysis and the procedures involved therein.

4.2 Evaluation metrics

We use the area under the receiver operating characteristic (ROC) curve (AUC) to measure and compare the performance of classifiers. According to (Huang & Ling, 2005), AUC is a better measure than accuracy when comparing the performance of classifiers. The ROC curve plots false positives (FP) vs. true positives (TP) using various threshold values. It compares the classifiers' performance across the entire range of class distributions and error costs (Huang & Ling, 2005).
Let N_L denote the total number of legitimate emails, and N_P denote the total number of phishing emails. Now, let $n_{L\to L}$ be the number of *legitimate* messages classified as *legitimate*,

$n_{L\to P}$ be the number of *legitimate* messages misclassified as *phishing*, $n_{P\to L}$ be the number of *phishing* messages misclassified as *legitimate*, and $n_{P\to P}$ be the number of *phishing* messages classified as *phishing*. False positives are legitimate emails that are classified as phishing, hence the false positive rate (FP) is denoted as:

$$FP = \frac{n_{L\to P}}{N_L}. \tag{8}$$

True positives are phishing emails that are classified as phishing, hence the true positive rate (TP) is denoted as:

$$TP = \frac{n_{P\to P}}{N_P}. \tag{9}$$

False negatives are phishing emails that are classified as legitimate, hence the false negative rate (FN) is denoted as:

$$FN = \frac{n_{P\to L}}{N_P}. \tag{10}$$

True negatives are legitimate emails that are classified as legitimate, hence the true negative rate (TN) is denoted as:

$$TN = \frac{n_{L\to L}}{N_L}. \tag{11}$$

Further we evaluate the predictive accuracy of classifiers, by applying the *weighted error* (W_{Err}) measure proposed in (Sakkis et al., 2003) and (Zhang et al., 2004). We test the classifiers using λ = 1 that is when legitimate and phishing emails are weighed equally. Hence the weighted accuracy (W_{Acc}), which is 1 - $W_{Err}(\lambda)$, can be calculated as follows

$$W_{Acc}(\lambda) = \frac{\lambda \cdot n_{L\to L} + n_{P\to P}}{\lambda \cdot N_L + N_P}. \tag{12}$$

In addition, we use several other measures to evaluate the performance of classifiers. We use the phishing *recall*(r), phishing *precision*(p), and phishing f_1 measures. According to (Sakkis et al., 2003), spam recall measures the percentage of spam messages that the filter manages to block (filter's effectiveness). Spam precision measures the degree to which the blocked messages are indeed spam (filter's safety). F-measure is the weighted harmonic mean of precision and recall. Here we use f_1 when recall and precision are evenly weighted. For the above measures, the following equations hold

$$p = \frac{n_{P\to P}}{n_{P\to P} + n_{L\to P}} \tag{13}$$

$$p = \frac{n_{P\to P}}{n_{P\to P} + n_{L\to P}} \tag{14}$$

$$f_1 = \frac{2pr}{p+r} \tag{15}$$

We use the AUC as the primary measure, as it allows us to gauge the trade off between the FP and TP rates at different cut-off points. Although the error rate W_{Err} (or accuracy) has been widely used in comparing classifiers' performance, it has been criticized as it highly depends on the probability of the threshold chosen to approximate the positive classes. Here we note that we assign new classes to the positive class if the probability of the class is greater than or equal to 0.5 (threshold=0.5). In addition, in (Huang & Ling, 2005) the authors prove theoretically and empirically that AUC is more accurate than accuracy to evaluate classi_ers' performance. Moreover, although classifiers might have different error rates, these rates may not be statistically significantly different. Therefore, we use the Wilcoxon signed-ranks test (Wilcoxon, 1945) to compare the error rates of classifiers and find whether the di_erences among these accuracies is significant or not (Demšar, 2006).

4.3 Experimental studies

We optimize the classifiers' performance by testing them using different input parameters. In order to find the maximum AUC, we test the classifiers using the complete dataset applying different input parameters. Also, we apply *10-fold-cross-validation* and average the estimates of all 10 folds (sub-samples) to evaluate the average error rate for each of the classifiers, using the 70 features and 6561 emails. We do not perform any preliminary variable selection since most classifiers discussed here can perform automatic variable selection. To be fair, we use L1-SVM and penalized LR, where variable selection is performed automatically.

We test NNet using different numbers of units in the hidden layer (i.e. different sizes (s)) ranging from 5 to 35. Further, we apply different weight decays (w) on the interconnections, ranging from 0.1 to 2.5. We find that a NNet with s = 35 and w = 0.7 achieves the maximum AUC of 98.80%.

RF is optimized by choosing the number of trees used. Specifically, the number of trees we consider in this experiment is between 30 and 500. When using 50 trees on our dataset, RF achieves the maximum AUC of 95.48%.

We use the L1-SVM C-Classification machine with radial basis function (RBF) kernels. L1-SVM can automatically select input variables by suppressing parameters of irrelevant variables to zero. To achieve the maximum AUC over different parameter values, we consider cost of constraints violation values (i.e. the "c" constant of the regularization term in the Lagrange formulation) between 1 and 16, and values of the γ parameter in the kernels between 1×10^{-8} and 2. We find that γ= 0.1 and c = 12 achieve the maximum AUC of 97.18%.

In LR we use penalized LR and apply different values of the lambda regularization parameter under the L2 norm, ranging from 1×10^{-8} to 0.01. In our dataset $\lambda = 1 \times 10^{-4}$ achieves the maximum AUC of 54.45%.

We use two BART models; the first is the original model and as usual, we refer to this as "BART". The second model is the one we modify so as to be applicable to classification, referred to as "CBART". We test both models using different numbers of trees ranging from 30 to 300. Also, we apply different power parameters for the tree prior, to specify the depth of the tree, ranging from 0.1 to 2.5. We find that BART with 300 trees and *power* = 2.5 achieves the maximum AUC of 97.31%. However, CBART achieves the maximum AUC of 99.19% when using 100 trees and *power* = 1.

4.4 Experimental results

In this section we present the experimental results be measuring the AUC using the complete dataset. In addition, we compare the *precision*, *recall*, *f*1, and W_{Err} measures using the optimum parameters achieved from the previous section. Figure 15 illustrates the ROCs for all classifiers.

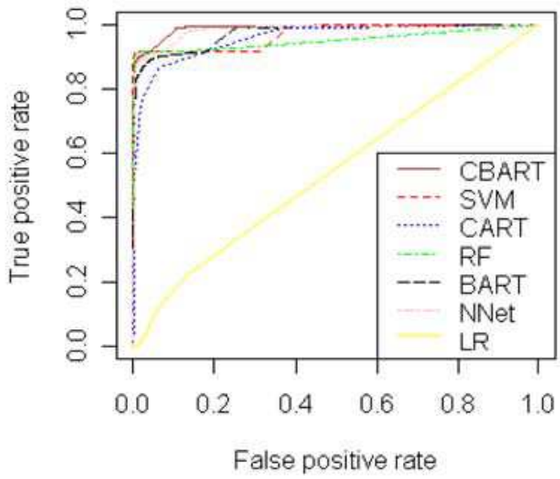


Fig. 15. ROC for all classifiers using the complete dataset.

Table 3 illustrates the AUC, FP, FN, *presicion*, *recall*, *f*1, and W_{Err} for all classi_ers. Note that the *FPrate* = 1 - *precision* and the *FNrate* = 1 - *recall*.

	AUC	W_{Err}	P	R	F1	FP	FN
CBART	99.19%	4.41%	97.02%	88.86%	92.8%	2.98%	11.14%
NNet	98.80%	4.31%	93.84%	85.68%	89.53%	6.16%	14.32%
SVM	97.18%	2.39%	94.57%	86.23%	90.17%	5.43%	13.77%
BART	97.31%	4.74%	93.82%	83.52%	88.32%	6.18%	16.48%
CART	96.06%	7.00%	88.45%	77.90%	82.67%	11.55%	22.10%
RF	95.48%	3.68%	95.75%	86.80%	91.02%	4.25%	13.20%
LR	54.45%	5.34%	92.71%	81.62%	86.77%	7.29%	18.38%

Table 3. Classifiers AUC, W_{Err}, precision, recall, f1, false positive, false negative.

In Table 4, we compare *p-value* of the error rate for each subsample among the 10 subsamples in cross validation by applying the Wilcoxon signed-rank test. Since CBART has a comparable error rate to that of RF, SVM, and NNet, we merely compare these three classifiers.

Classifier	p-value
RF	0.03711
SVM	0.1602
NNet	0.625

Table 4. Comparing the p-value using the Wilcoxon-signed ranked test

4.5 Discussion

Here we investigate the application of a modified version of BART for phishing detection. The results demonstrate that CBART outperforms other classifiers on the phishing dataset,

achieving the maximum AUC of 99.19%. The results show that CBART has the highest AUC, followed by NNet with 98.80%, BART with 97.31%, SVM with 97.18%, CART with 96.06%, RF with 95.48%, and LR with 54.45% respectively. Apparently the AUC for CBART has improved by 1.88% compared to the original BART model.

The results show that CBART's error rate is comparable to other classifiers (merely RF, SVM, and NNet). When the error rates for the 10 subsamples are compared using the Wilcoxon-singed ranked test, the *p-value* of the tests for SVM and NNet are greater than 0.05, which is an indication that the difference in the results is not statistically significantly different (see Table 4). However, for RF, the *p-value* is less than 0.05, which is an indication that the error rates *may be* statistically significantly different. Table 3 illustrates that the AUC for CBART is greater than RF by approximately 4%, therefore, we conclude that CBART's performance is better than RF.

SVM achieves the minimum error rate (maximum accuracy) of 2.39%, followed by RF with 3.68%, NNet with 4.31%, CBART with 4.41%, BART with 4.74%, LR with 5.34%, and CART with 7% respectively. Note that the accuracy of CBART has improved, insignificantly though, by 0.33%.

CBART achieves the minimum FP rate of 2.98%, followed by RF with 4.24%, SVM with 5.43%, NNet with 6.16%, BART with 6.18%, LR with 7.29%, and CART with 11.55%. The minimum FN rate is achieved by CBART with 11.14%, followed by RF with 13.20%, SVM with 13.77%, NNet 14.32%, BART 16.48%, LR 18.38%, and CART 22.10% respectively.

It is well known that LR performs very well when the relationship underlying data is linear. We believe that the comparatively low predictive accuracy of LR is an indication of a non-linear relationship among the features and the response in the dataset.

With its superior classification performance, relatively high predictive accuracy, relatively low FP rate, and distinguished features, CBART proves to be a competitive and a practical method for phishing detection.

5. Conclusions

A modified version of Bayesian Additive Regression Trees (CBART) proved to be suitable as a phishing detection technique. The performance of CBART was compared against well-known classification methods on a phishing dataset with both textual and structural features.

The results showed that CBART is a promising technique for phishing detection and it has several features that make it competitive. Further, the results demonstrated that the performance of the modified BART model outperforms other well-known classifiers and comparatively achieves low error rate and false positives. CBART outperformed all the other classifiers and achieved the maximum AUC of 99.19% on the phishing dataset we built, decreasing by 1.88% compared to AUC of BART prior to the modification. SVM achieved the minimum error rate of 2.39% leaving behind, RF with 3.68% and NNet with 4.31% followed by CBART with 4.41%. In addition, CBART achieved the minimum FP rate of 2.98% followed by RF with 4.25%, SVM with 5.43%, NNet with 6.16%, and BART with 6.18%.

Automatic variable selection in BART motivates future work to explore BART as a variable selection technique. This includes comparing its performance to other well known variable selection methods.

6. References

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit* (pp. 60-69). New York, NY, USA: ACM.

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2008). Bayesian additive regression trees-based spam detection for enhanced email privacy. *ARES '08: Proceedings of the 3rd International Conference on Availability, Reliability and Security* (pp. 1044-1051).

Albert, J. H., & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, *88*, 669-679.

Berry, M. W. (Ed.). (2004). *Survey of text mining: Clustering, classification, and retrieval*. Springer.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5-32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Chapman & Hall/CRC.

Chandrasekaran, M., Narayanan, K., & Upadhyaya, S. (2006). Phishing email detection based on structural properties. *NYS Cyber Security Conference*.

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, *93*, 935-947.

Chipman, H. A., George, E. I., & McCulloch, R. E. (2006). BART: Bayesian Additive Regression Trees. http://faculty.chicagogsb.edu/robert.mcculloch/research/code/BART-7-05.pdf.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1-30.

Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to detect phishing emails. *WWW '07: Proceedings of the 16th international conference on World Wide Web* (pp. 649-656). New York, NY, USA: ACM Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning data mining, inference, and prediction*. Springer Series in Statistics. Springer.

Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *17*.

James, L. (2005). *Phishing exposed*. Syngress.

Kandola, J. S. (2001). *Interpretable modelling with sparse kernels*. Doctoral dissertation, University of Southampton.

Marques de Sa, J. P. (2001). *Pattern recognition: Concepts, methods and applications*. Springer.

Massey, B., Thomure, M., Budrevich, R., & Long, S. (2003). Learning spam: Simple techniques for freely-available software. *USENIX Annual Technical Conference, FREENIX Track* (pp. 63-76).

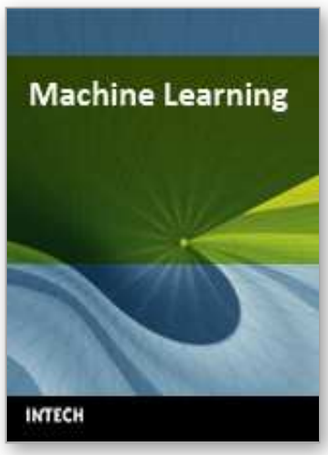
Nazario, J. (2007). Phishing corpus. http://monkey.org/~jose/phishing/phishing3.mbox.

Neal, R. M. (1995). *Bayesian learning for neural networks*. Springer-Verlag Publishers.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C., & Stamatopoulos, P. (2003). A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, *6*, 49-73.

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill. Weiss, S., Indurkhya, N., Zhang, T., & Damerau, F. (2004). *Text mining: Predictive methods for analyzing unstructured information*. Springer.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, *1*, 80-83.
Zhang, L., Zhu, J., & Yao, T. (2004). An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, *3*, 243-269.



Machine Learning

Edited by Abdelhamid Mellouk and Abdennacer Chebira

ISBN 978-953-7619-56-1
Hard cover, 450 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

Machine Learning can be defined in various ways related to a scientific domain concerned with the design and development of theoretical and implementation tools that allow building systems with some Human Like intelligent behavior. Machine learning addresses more specifically the ability to improve automatically through experience.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang and Suku Nair (2009). Hardening Email Security via Bayesian Additive Regression Trees, Machine Learning, Abdelhamid Mellouk and Abdennacer Chebira (Ed.), ISBN: 978-953-7619-56-1, InTech, Available from: http://www.intechopen.com/books/machine_learning/hardening_email_security_via_bayesian_additive_regression_trees

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen