A Comparison of Machine Learning Techniques for Phishing Detection

Saeed Abu-Nimeh[1], Dario Nappa[2], Xinlei Wang[2], and Suku Nair[1]

SMU HACNet Lab
Southern Methodist University
Dallas,TX 75275
[1]{sabunime,nair}@engr.smu.edu
[2]{dnappa,swang}@smu.edu

ABSTRACT

There are many applications available for phishing detection. However, unlike predicting spam, there are only few studies that compare machine learning techniques in predicting phishing. The present study compares the predictive accuracy of several machine learning methods including Logistic Regression (LR), Classification and Regression Trees (CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random Forests (RF), and Neural Networks (NNet) for predicting phishing emails. A data set of 2889 phishing and legitimate emails is used in the comparative study. In addition, 43 features are used to train and test the classifiers.

Keywords

BART, CART, classification, logistic regression, machine learning, NNet, phishing, random forests, SVM

1. INTRODUCTION

There is no agreed upon definition for phishing. However, most definitions agree that the goal of a phishing scam is to steal individuals' personal confidential information [3, 11, 17]. The media of the attack may vary depending on the attack setup. For instance, *Pharming* is a type of phishing, where the attacker misdirects users to fraudulent sites or proxy servers, typically through Domain Name System (DNS) hijacking or poisoning [3]. In this case an attacker can steal victims' information by acquiring a domain name for a website and redirecting that website's traffic to phishing website without sending forged emails. Nevertheless, email remains the most favorable vehicle for phishing. The abundance of off-the-shelf bulk mailing tools (dubbed as *mailers*) simplifies the job of phishers and help in delivering a huge number of emails to a large number of victims.

Studies show a steady increase in phishing activities as well as the related cost. In 2003 direct phishing-related loss to US banks and credit card issuers was estimated by $1.2 billion which grew to $2 billion in 2005. In January 2007, the total number of unique phishing reports submitted to the Anti-Phishing Working Group (APWG) was 29,930. This is the highest number of reports recorded by the APWG [3]. Compared to the previous peak in June 2006, the number of submitted reports increased by 5%. Even though there are several solutions proposed and implemented for detection and prevention of phishing attacks most of them suffer from unacceptable levels of false positives or missed detection.

The contribution of this study is to compare the predictive accuracy of six classifiers on a phishing data set. The classifiers include Logistic Regression (LR), Classification and Regression Trees (CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random Forests (RF), and Neural Networks (NNet). A data set is constructed from 1171 raw phishing emails and 1718 legitimate emails. In addition, 43 features (variables) are used in training and testing the classifiers.

The rest of the paper is organized as follows: in Section 2 we discuss related work. In Section 3 we illustrate the classification methods used in the study. In Section 4 we discuss the data set construction, evaluation metrics, and the preliminary setup. In Section 5 we demonstrate our experimental studies. The results are presented in Section 6 and discussed in Section 7. We draw conclusions and motivate for future work in Section 8.

2. RELATED WORK

According to the APWG there are, in general, three main categories of phishing and fraud defense mechanisms; detective, preventive, and corrective solutions [3]. These categories and their corresponding subcategories are summarized in Table 1.

In the following we provide a brief description of few available phishing detection techniques. First, we describe anti-phishing toolbars. Afterwards, we review two research studies that apply machine learning in phishing detection.

2.1 Anti-Phishing Toolbars

Anti-phishing toolbars are ubiquitously available and commonly used by naive or non-technical computer users to help alleviate the phishing problem. Although these toolbars help mitigate the problem, many research studies have demonstrated the ineffectiveness of such techniques. One of

APWG eCrime Researchers Summit, October 4-5, 2007, Pittsburgh, PA, USA.

Table 1: Phishing and fraud solutions.

Detective solutions	Preventive solutions	Corrective solutions
Monitors account life cycle	Authentication	Site takedown
Brand monitoring	Patch and change management	Forensics and investigation
Disables web duplication	Email authentication	
Performs content filtering	Web application security	
Anti-Malware		
Anti-Spam		

the major problems with such solution is that, quite often the spoofed link is tested without any consideration to the context in which it was presented to the user thereby losing accuracy. Another problem is that once the user enters the address of the phishing site in the browser address bar, the user is exposed immediately to any attack carried by the site.

Wu et al. [23] evaluated the effectiveness of security toolbars in preventing phishing attacks. They performed experiments on three security toolbars, the browsers address bar, and the status bar. A total of 30 subjects were included in experiments. They showed that all tested security toolbars were ineffective in preventing phishing attacks. Users were spoofed 34% of the time. 20 out of 30 users got spoofed by at least one phishing attack. 85% of the spoofed users thought that websites look legitimate or exactly the same as they visited before. 40% of the spoofed users were tricked because of poorly designed websites, especially when using improper redirections. In consequence, the study concludes that there are two main reasons behind the users failing into these attacks: First, users discarded the toolbar display, as the content of web pages looks legitimate or professional. Second, many companies do not follow good practice in designing their websites and the toolbar cannot help users to distinguish poorly designed websites from malicious phishing attacks.

In another study [10], the authors tested the effectiveness of 10 security toolbars and found that three of the 10 toolbars were able to identify over 75% of the phishing sites tested. However, four of the toolbars were not able to identify 50% of the tested sites. This shows that there is a problem in the design of these solutions and more work needs to done to improve the quality of results.

2.2 Machine Learning Techniques

Most of the machine learning algorithms discussed here are categorized as *supervised* machine learning. That is where an algorithm (classifier) tries 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). In the case of phishing classification, a classifier will try to classify an email to phishing or legitimate (response) by learning certain characteristics (features) in the email. In the following we summarize two research studies that apply machine learning in phishing classification.

Chandrasekaran et al. [7] 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. [13] 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 data set 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.

In our study we use a relatively new corpus of emails, which represents the newest trends in phishing emails. In addition, the two techniques mentioned above rely on a training data set with a low number of features, namely 10 and 25. We note that omitting important features in training hurts the learning performance, and on the other hand including redundant ones leads to over-fitting. Furthermore, we use more evaluation measures in our comparison than in [7] and [13]. In addition to the measures used in the studies above, we apply cost-sensitive measures to penalize false positives and we compare the area under the ROC curve as well.

3. STUDIED METHODS FOR PHISHING DETECTION

In the following subsections, we briefly present the classification methods that are used in our comparative study.

3.1 Logistic Regression

Logistic regression is the most widely used statistical model in many fields for binary data (0/1 response) prediction. It has been widely applied 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.

3.2 Classification and Regression Trees

CART or Classification and Regression Trees [6] 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. nonlinear 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 [8].

CART is flexible 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 Random Forests

Random forests are classifiers 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 [5]. 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 $k \ll p$ 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.

3.4 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 [18]. 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 1 depicts an example of neural network. The neural

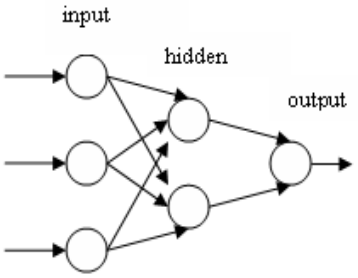

Figure 1: Neural Network.

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 significant 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 [19]

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

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.

3.5 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 2 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 [18].

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.

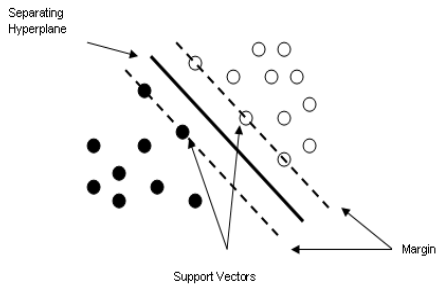

Figure 2: Support Vector Machines.

3.6 Bayesian Additive Regression Trees

Bayesian Additive Regression Trees (BART) is a new technique developed by [9]. The method is used to discover 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 regression tree with arbitrary structure, and contributes a small amount to the overall model as a *weak learner*, when m is chosen large. If we consider g_i in (1) as a tree model, then we can write

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

where T_i is the binary tree model, example of which is given by Figure 3. M_i is the vector containing terminal node parameters (μ's), m is the total number of trees, and σ^2 is the variance of noise. By using a sum-of-trees model, BART is

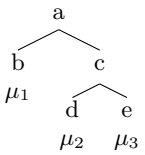

Figure 3: Binary tree model.

vastly more flexible than single tree models that can hardly account for additive effects. Each tree component is actually a "weak learner", which explains a small and different part of the unknown relationship between the input and output. In addition, BART can easily incorporate high-order interaction effects among three or more input variables, which can be difficult to be captured by other additive models. Furthermore, BART does not require variables selection, which is preformed automatically as the trees are built.

The bayesian approach usually brings heavy computation time due to its nature. Specifying a *prior* for the parameters of the model and thus predicting the *posterior* probability requires complex computations. Generally, the process of finding the posterior is approximated by Markov chain Monte Carlo (MCMC) simulation.

4. EVALUATION APPROACH

In this section we describe how we construct the testing data set from the raw phishing emails. In addition, we describe the evaluation metrics we use in the comparison. Lastly, we describe the preliminary setup of the experiments.

4.1 Data Set Description

Similar to the "spam email database" donated by Forman and created by Hopkins et al. [22], we construct a phishing data set by processing a set of 1171 raw phishing emails collected between November 15, 2005 and 7 August, 2006 [21]. This set of phishing emails covers many of the newer trends in phishing. For the legitimate portion of the data set, we use 1718 messages collected from our own mailboxes. Hence in total our data set contains 2889 emails, 59.5% of them are legitimate. The percentage of legitimate emails is approximately the same as used in [22].

Each email is converted into a vector $\vec{x} = \langle x_1, x_2, \cdots, x_p \rangle$, where $x_1, \cdots, x_p$ are the values corresponding to a specific feature (variable) [14]. The data set consists of 43 continuous features (variables) and one binary response variable, which indicates that email is phishing=1 or legitimate=0. The features represent the frequency of the most frequent terms that appear in phishing and legitimate emails. Choosing words (terms) as features is widely applied in the text mining literature and is referred to as a "bag-of-words". In the following we describe the steps of constructing the phishing data set.

We start by striping all *attachments* from emails in order to facilitate the process of text mining the email body. Then, we extract the *header* information of all emails keeping the email subject and body. Afterwards, we extract the *html tags* and *elements* from the body of the emails, leaving out the body as plain text. Thus, we filter out *stopwords* from the text of the body. We use a list of 424 commonly used English stopwords. Then, we perform *stemming* by removing the suffixes from words to get their common origin. For example, "suspended" and "suspending" are converted to their origin "suspend" [4]. Lastly, we find the most frequent terms using TF-IDF (Term Frequency Inverse Document Frequency) and choose the most frequent terms that appear in the corpus. 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 [4].

4.2 Evaluation Metrics

Following previous spam classification research, we use the spam $recall(r)$, spam $precision(p)$, and spam f_1 measures. According to [2], 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, as recall and precision are evenly weighted.

Let $n_{L \to L}$ be the number of *legitimate* messages classified as *legitimate*, $n_{L \to S}$ be the number of *legitimate* messages misclassified as *spam*, $n_{S \to L}$ be the number of *spam* messages misclassified as *legitimate*, and $n_{S \to S}$ be the number of *spam* messages classified as *spam*. Thus the following

equations hold

$$r = \frac{n_{S\to S}}{n_{S\to S} + n_{S\to L}} \tag{3}$$

$$p = \frac{n_{S\to S}}{n_{S\to S} + n_{L\to S}} \tag{4}$$

$$f_1 = \frac{2pr}{p+r} \tag{5}$$

Generally, false positives are described as normal elements that are identified as anomalous; however, false negatives are anomalous elements that are identified as normal. In addition to the previous metrics we use cost-sensitive measures namely the "weighted accuracy" and the "weighted error" measures proposed in [1] and [25]. According to the authors, these metrics are used to assign different weights on false positive and false negative errors. Let N_L denote the total number of legitimate emails, and N_S denote the total number of spam emails. The authors apply different λ values to assign different weights on the accuracy or error. Specifically, they apply $\lambda = 1$ when legitimate and spam emails are weighed equally. However, when $\lambda = 9$, false positives are penalized nine times more than false negatives and when $\lambda = 999$ false positives are as bad as 999 false negatives. Simply said the higher the λ the more penalty put on false positives. [25].
Now, the weighted accuracy (W_{Acc}) is calculated as follows

$$W_{Acc}(\lambda) = \frac{\lambda \cdot n_{L\to L} + n_{S\to S}}{\lambda \cdot N_L + N_S} \tag{6}$$

Hence, the "weighted error" (W_{Err}) is

$$W_{Err}(\lambda) = 1 - W_{Acc}(\lambda) \tag{7}$$

4.3 Experimental Setup

In our experiments we use all 43 variables in the data set. Further, we perform *10-fold-cross-validation*. Cross validation is a method to estimate the error rate efficiently and in unbiased way. The procedure works as follows: the data set is divided into k sub-samples (in our experiments $k = 10$). A single sub-sample is chosen as testing data, and the remaining $k-1$ sub-samples are used as training data. The procedure is repeated k times, in which each of the k sub-samples is used exactly once as the testing data. All results are averaged and a single estimation is provided [20].

We preform 30 simulations (i.e. experiments are repeated 30 times) and thus their output is averaged to compare the results. Also, we note that we exclude *Naive Bayes* as one of the candidates for the studied methods because of its poor prediction rate on the studied data set. The error rate for the naive bayes classifier is around 40%, which is very high compared to the other methods. We expect that this due to using all 43 variables in training and testing.

We note that BART was not originally designed for classification. BART is a learner to predict quantitative outcomes from observations via regression. There is a distinction between regression and classification. 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. In order to enable BART to adapt to classification problems, we use a threshold value (th) to approximate the prediction output. Simply, the continuous output is rounded to 0 or 1 based on the th value we choose. We perform several experiments to find the th value that provides the lowest error rate. Thus use this value though out our experiments.

In LR we use a logistic transformation and a binomial distributed error. In CART we use one classification tree for classification. We use a standard non-linear feed-forward NNet with a sigmoid activation function.

5. EXPERIMENTAL STUDIES

In this section we demonstrate our experimental studies to investigate the predictive accuracy of NNet, LR, RF, BART, CART, and SVM.

In order to find the minimum average error rate for NNet, we test it using different number of units in the hidden layer (different sizes), namely 5, 10, 15, and 20 units. Additionally, we apply different weight decays on the interconnections, namely 0.1, 0.2, 0.3, 0.4, 0.5, 1, 1.5, 2, and 2.5 decays. The experiments show that a NNet with size 10 and weight decay of 0.1 provides the lowest error rate, which is 0.1161. Thereafter, we use this very NNet in the evaluation against other methods. Figure 4 depicts the different weight decays and the error rate using different sizes of the NNet.

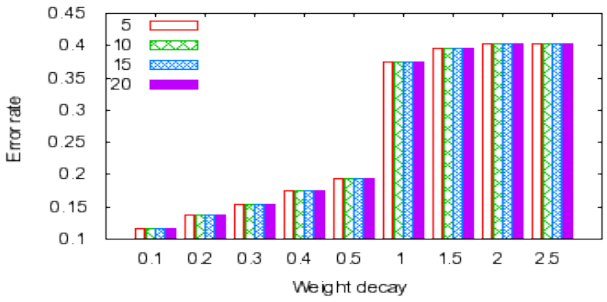

Figure 4: Error rate in NNet with different sizes and different weight decays.

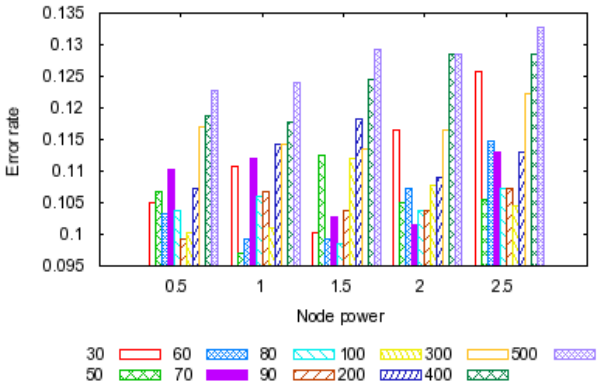

Figure 5: Error rate using BART with different power parameters and different tree sizes.

Furthermore, we find the minimum average error rate for BART and hence we test BART using different number of

trees, namely 30, 50, 60, 70, 80, 90, 100, 200, 300, 400, and 500 trees. Moreover, BART sets parameters to specify the depth of the tree (or power parameter for tree prior, see [9]), therefore we test it using different parameters namely, 0.5, 1, 1.5, 2, and 2.5. Further, we simulate 1000 MCMC simulations and specify 100 MCMC "burn-in" iterations, so the *posterior* draws are done in 900 simulations. The experiments show that BART with 50 trees and power parameter equals to 1 provides the lowest error rate ($W_{Err} = 0.0969$). Subsequently, we use this very BART setup in the evaluation verses other methods. Figure 5 depicts the different power parameters and the error rate using different number of trees in BART.

To find the minimum average error rate for RF, we test it using different number of trees, namely 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, and 500 trees. The experiments show that a RF with 50 trees has the minimum error rate ($W_{Err} = 0.0775$). Subsequently, we use this very RF setup in the evaluation verses other methods. Figure 6 depicts the error rate using different number of trees in RF.

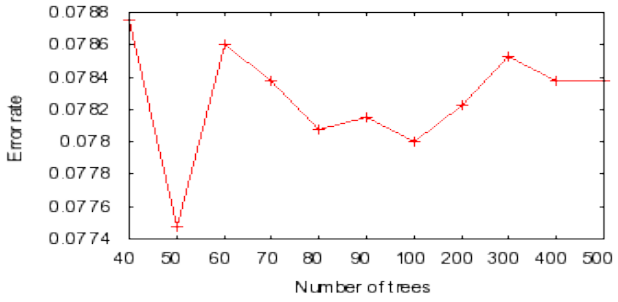

Figure 6: Error rate in RF using different number of trees.

Now, we test all methods using 10-fold-cross-validation and we average the error rate of all sub-samples (10 folds) to get the weighted error rate (W_{Err}) used in section 6. Note that we apply cost-sensitive measures to calculate W_{Err} when $\lambda = 1$ (i.e legitimate and phishing emails are weighed equally) and $\lambda = 9$ (i.e. false positives are penalized nine times more than false negatives). Figure 7 shows the error rate when $\lambda = 1$ and Figure 8 shows the error rate when $\lambda = 9$ for all 10 sub-samples during cross validation.

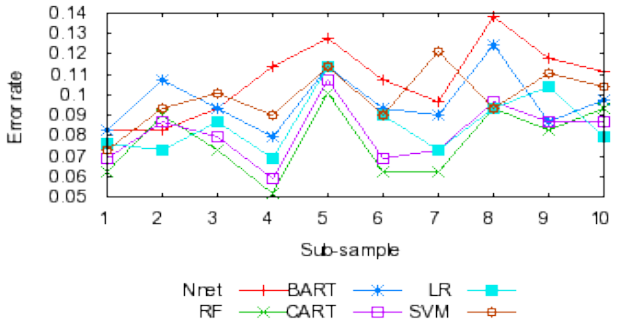

Figure 7: $W_{Err}(\lambda = 1)$ for all classifiers of each sub-sample during the 10-fold-cross-validation.

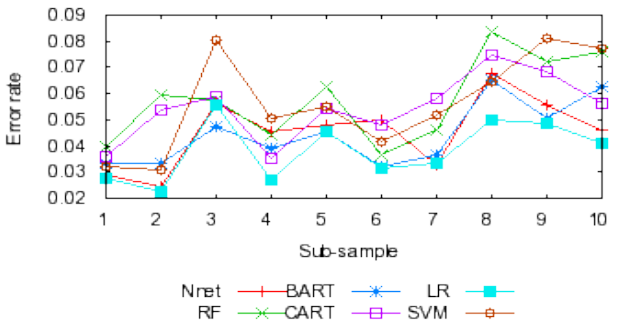

Figure 8: $W_{Err}(\lambda = 9)$ for all classifiers of each sub-sample during the 10-fold-cross-validation.

In the next section we demonstrate the results of our experiments.

6. EXPERIMENTAL RESULTS

As we mentioned in the previous section, to find the error rate for each classifier we calculate the average error rate in all sub-samples during cross validation. Figure 9 shows the average error rate for all classifiers when $\lambda = 1$.

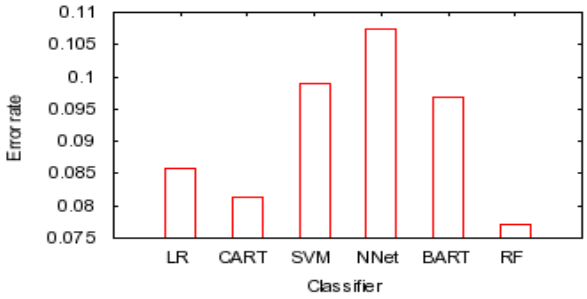

Figure 9: Average $W_{Err}(\lambda = 1)$ for all classifiers.

The W_{Err} when $\lambda = 9$ for all classifiers is calculated similarly by taking the average of all sub-samples during cross validation. Figure 10 shows the average error rate for all classifiers when $\lambda = 9$. It is noticeable that the average error rate changes after penalizing false positives more than false negatives.

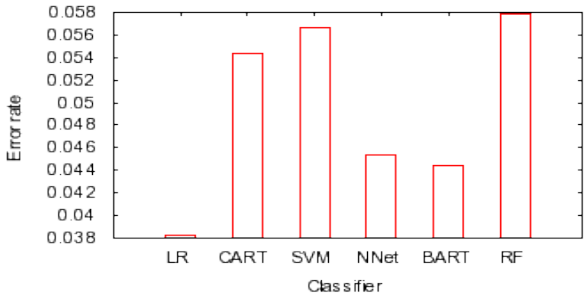

Figure 10: Average $W_{Err}(\lambda = 9)$ for all classifiers.

In phishing classification, false positives (FP) are legitimate emails that are misclassified as phishing and false negatives (FN) are phishing emails that are misclassified as legitimate. Table 2 summarizes the false positive rate and the false negative rate for all classifiers.

Table 2: *False positive (FP)* rate, and *false negative (FN)* rate for all classifiers

	FP	***FN***
LR	04.89%	17.04%
CART	07.68%	12.93%
SVM	07.92%	17.26%
NNet	05.85%	21.72%
BART	05.82%	18.92%
RF	08.29%	11.12%

The receiver operating characteristic (ROC) curve is a graph to plot sensitivity vs. specificity for a binary classifier, in which various threshold values are compared. Sensitivity, in our case, is the probability of predicting phishing given true state is phishing. However, specificity is the probability of predicting legitimate email given true state is legitimate [12]. Usually, the ROC curve is presented by plotting false positives (FP) vs. true positives (TP) (i.e. phishing emails detected as phishing) and this comparison is what we use here. In Figure 11 we compare the area under the ROC curve (AUC) for all classifiers.

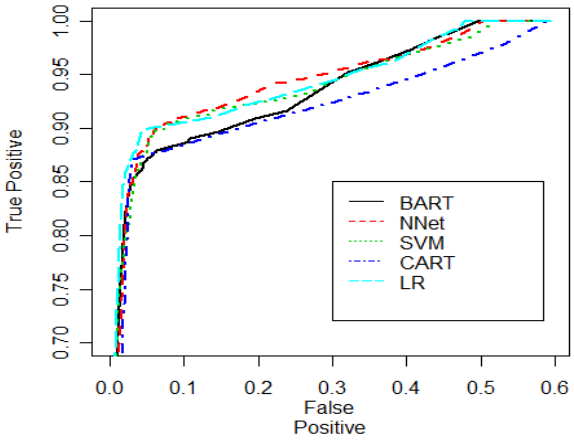

Figure 11: ROC for all classifiers.

Lastly, we compare the *precision*, *recall*, and f_1 for all classifiers in Table 3.

7. DISCUSSION

The study compares the predictive accuracy of several classifiers for predicting phishing emails. A data set comprising of 2889 phishing and legitimate emails is examined. The experiments show that RF has the lowest W_{Err} of 07.72%, when legitimate and phishing emails are weighted equally ($\lambda = 1$), followed by CART 08.13%, followed by LR 08.58%, followed by BART 09.69%, then SVM 09.90%, and finally NNet with 10.73% (see Figure 9).

Table 3: Comparison of *precision*, *recall*, and f_1

	Precision	***Recall***	***F*$_1$**
LR	95.11%	82.96%	88.59%
CART	92.32%	87.07%	89.59%
SVM	92.08%	82.74%	87.07%
NNet	94.15%	78.28%	85.45%
BART	94.18%	81.08%	87.09%
RF	91.71%	88.88%	90.24%

In email filtering false positives and false negatives are given more consideration when studying the performance (predictive accuracy) of a classifier. That is because false positives are more expensive than false negatives in the real world. Now, despite the fact that RF outperforms all classifiers and has the lowest prediction error, it suffers from the highest false positive rate (see Table 2). RF misclassifies 08.29% of legitimate emails as phishing, followed by SVM 07.92%, followed CART 07.68%, followed by NNet 05.85%, then BART 05.82%, and finally LR 04.89%. As we mentioned earlier, practically, false positives are more expensive than false negatives. Users do not like their legitimate message, which might be important, to be deleted or misclassified as junk. In consequence, LR might be more preferable in practice than others, as it has the lowest false positive rate. RF has the lowest false negative rate of 11.12%, followed by CART 12.93%, followed by LR 17.04%, followed by SVM 17.26%, then BART 18.92%. NNet has the highest false negatives of 21.72%.

In order to find a measure that combines the prediction error of a classifier with the false positive rate, we apply cost-sensitive measures on the error rate (i.e. use weighted error). Obviously, the error rate changes when we apply cost-sensitive measures and penalize false positives 9 times more than false negatives ($\lambda = 9$). Figure 10 shows that LR, now, outperforms all classifiers with a weighted error rate of 03.82%, followed by BART 04.45%, followed by NNet 04.54%, followed by CART 05.43%, then SVM 05.66%, and finally RF 05.78%. Note that RF performs the worst when penalizing false positives. This is indeed supported by the results shown in Table 2, as RF achieves the worst false positive rate among all classifiers.

Another measure that can be used to compare the performance of classifiers is the area under the ORC curve (AUC). In general, the AUC provides more accurate comparison in regards to false positives, as it depicts false positives vs. true positives. Table 4 summarizes the area under the curve for all classifiers. It is obvious that NNet has the highest AUC of 0.9448 (also see Figure 11), followed by RF 0.9442, followed by SVM 0.9403, followed by LR 0.9388, followed by BART 0.9324, and followed by CART 0.9145.

The results show that it is difficult to find an evaluation metric which summarizes all aspects of comparison. The error rate solely does not provide insight into false positives and false negatives. This is similar to what is found by Hand [15]. The author shows that using the error rate, without applying costs, leads to inaccurate comparisons. Also, he ar-

Table 4: Area under the ROC curve (AUC) for all classifiers

	AUC
LR	0.9388
CART	0.9145
SVM	0.9403
NNet	0.9448
BART	0.9324
RF	0.9442

gues that using straightforward error rate as a performance measure when comparing classifiers' performance is misleading. Further, he shows that variable misclassification costs can not be ignored in real-world problems.

Similarly, the AUC depends on the probability of the threshold chosen to plot the curve. Therefore, relying only on the AUC may be an inefficient measure to study the predictive accuracy of classifiers (see [16]). In conclusion, researches can choose to apply measures that best fit their problems. We believe that email classification in general, be it phishing or spam classification, requires more penalty on false positives, as they are more expensive in the real world. Consequently, using cost-sensitive measures such as the weighted error is suggested.

8. CONCLUSIONS AND FUTURE WORK

In the present study we investigated the predictive accuracy of six classifiers on a phishing data set. The classifiers included Logistic Regression (LR), Classification and Regression Trees (CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random Forests (RF), and Neural Networks (NNet). We constructed a data set from 1171 raw phishing emails and 1718 legitimate emails, where 43 features were trained and tested to predict phishing emails. During training and testing we used 10-fold-cross-validation and averaged the estimates of all 10 folds (sub-samples) to evaluate the mean error rate for all classifiers.

The results showed that, when legitimate and phishing emails are weighted equally, RF outperforms all other classifiers with an error rate of 07.72%, followed by CART, LR, BART, SVM, and NNet respectively. NNet achieved the worst error rate of 10.73%. Although RF outperformed all classifiers, it achieved the worst false positive rate of 08.29%. LR had the minimum false positive rate of 4.89%. When applying cost-sensitive measures, i.e. penalizing false positives 9 times more than false negatives, LR outperformed all classifiers achieving the minimum weighted error rate of 03.82% followed by BART, NNet, CART, SVM, and RF respectively. However, RF had the worst weighted error rate of 05.78% when $\lambda = 9$. Further, we compared the area under the ROC curve (AUC) for all classifiers. NNet had the highest AUC of 0.9448, followed by RF, SVM, LR, BART, and CART respectively. Furthermore, we compared false negative, precision, recall and f-1 rates for all classifiers.

We argued that the error rate solely (i.e. penalizing phishing and legitimate emails equally) does not provide insight into false positives. Also, applying the AUC as a measure by itself might be inefficient to study predictive accuracy of a classifier. We suggested using cost-sensitive measures to provide more conclusive results about the predictive accuracy of classifiers.

The results motivate future work to explore inclusion of additional variables to the data set, which *might* improve the predictive accuracy of classifiers. For instance, analyzing email headers has proved to improve the prediction capability and decrease the misclassification rate of classifiers [24]. Further, we will explore adding the features used in [7] and [13] to our data set and will study their effect on classifiers' performance. In addition, we will explore developing an automated mechanism to extract new features from raw phishing emails in order to keep up with new trends in phishing attacks.

9. ACKNOWLEDGMENTS

The authors would like to thank Jose Nazario for providing the phishing corpus of raw emails. Also, we would like to thank the anonymous referees for their helpful comments and reviews.

10. REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proc. of the workshop on Machine Learning in the New Information Age*, 2000.

[2] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167, New York, NY, USA, 2000. ACM Press.

[3] Anti-Phishing Working Group. http://www.antiphishing.org/.

[4] M. W. Berry, editor. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer, 2004.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[6] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.

[7] M. Chandrasekaran, K. Narayanan, and S. Upadhyaya. Phishing email detection based on structural properties. In *NYS Cyber Security Conference*, 2006.

[8] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–947, 1998.

[9] H. A. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian Additive Regression Trees. *Journal of the Royal Statistical Society*, 2006. Ser. B, Revised.

[10] L. F. Cranor, S. Egelman, J. Hong, and Y. Zhang. Phinding phish: An evaluation of anti-phishing toolbars. Technical Report CMU-CyLab-06-018, CMU, November 2006.

[11] A. Emigh. Online identity theft: Phishing technology, chokepoints and countermeasures. Technical report, Radix Labs, 2005.

[12] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[13] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 649–656, New York, NY, USA, 2007. ACM Press.

[14] S. G and M. MJ. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[15] D. J. Hand. Classifier technology and the illusion of progress. *Statistical Science*, 21(1):1–15, 2006.

[16] F. E. J. Harrell. *Regression Modeling Stratigies*. Springer, 2001.

[17] L. James. *Phishing Exposed*. Syngress, 2005.

[18] J. P. Marques de Sa. *Pattern Recognition: Concepts, Methods and Applications*. Springer, 2001.

[19] B. Massey, M. Thomure, R. Budrevich, and S. Long. Learning spam: Simple techniques for freely-available software. In *USENIX Annual Technical Conference, FREENIX Track*, pages 63–76, 2003.

[20] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[21] J. Nazario. Phishing corpus. http://monkey.org/ jose/phishing/phishing2.mbox.

[22] Spambase. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/.

[23] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2006.

[24] L. Zhang and T. Yao. Filtering junk mail with a maximum entropy model. In *Proceeding of 20th International Conference on Computer Processing of Oriental Languages (ICCPOL03)*, pages 446–453, 2003.

[25] L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.

APPENDIX

A. DATA SET STATISTICS

Table 5 shows the variables in the data set and their characteristics.

Table 5: Variables and characteristics of the data set

Variable	Min.	Max.	Avg.	Stdev.	Coeff. var. (%)
1	0	0.7	0.00452	0.02863	633.5
2	0	0.14	0.01656	0.06783	409.6
3	0	0.66667	0.00151	0.01679	1112.4
4	0	0.5	0.00313	0.01693	541.4
5	0	0.26	0.00816	0.08964	1098.7
6	0	0.127273	0.0104	0.05491	528
7	0	0.5	0.00084	0.01483	1767.3
8	0	0.3	0.00194	0.01268	652.9
9	0	0.27778	0.00038	0.00599	1568.5
10	0	0.27778	0.00033	0.00584	1754.5
11	0	0.1	0.00337	0.03066	910.7
12	0	0.66667	0.00206	0.0174	846.1
13	0	0.6	0.00032	0.01175	3640.6
14	0	0.10256	0.00039	0.00415	1053.5
15	0	0.2	0.00054	0.0057	1054.2
16	0	0.1	0.00015	0.00251	1638.4
17	0	0.20588	0.00017	0.00423	2436
18	0	0.12195	0.00012	0.00304	2617.7
19	0	0.09091	0.00102	0.00469	459.3
20	0	0.25	0.00009	0.00466	5119.7
21	0	0.11111	0.0002	0.00365	1819.5
22	0	0.54545	0.00099	0.01223	1235.5
23	0	0.54545	0.00082	0.01249	1528.4
24	0	0.23529	0.00034	0.00569	1691.7
25	0	0.20588	0.00009	0.00389	4513.5
26	0	0.05556	0.00012	0.00178	1496.3
27	0	0.05556	0.00008	0.00138	1771.7
28	0	0.2	0.00018	0.00424	2392.8
29	0	0.03311	0.00006	0.00128	2035.5
30	0	0.10811	0.00009	0.00216	2480.5
31	0	0.3	0.00113	0.01058	939.3
32	0	0.3	0.00089	0.00964	1077.6
33	0	0.15	0.00059	0.00531	905.5
34	0	0.05263	0.00021	0.00179	851.2
35	0	0.04688	0.00041	0.00228	555
36	0	0.05263	0.00029	0.00176	601.2
37	0	0.02304	0.0002	0.0011	550.6
38	0	0.02344	0.00019	0.00114	612.5
39	0	0.04082	0.0002	0.00134	665
40	0	0.07895	0.00042	0.00241	576.2
41	0	0.00792	0.00007	0.00046	680.7
42	0	0.05263	0.00008	0.00119	1567.1
43	0	0.05556	0.00056	0.00272	486.7
44	0	1	0.40498	0.49089	121.2