A Distributed Architecture for Phishing Detection using Bayesian Additive Regression Trees

Saeed Abu-Nimeh[1], Dario Nappa[2], Xinlei Wang[2], Suku Nair[1]
SMU HACNet Lab
Southern Methodist University
Dallas, TX 75275
[1]{sabunime,nair}@engr.smu.edu
[2]{dnappa,swang}@smu.edu

***Abstract*—With the variety of applications in mobile devices, such devices are no longer deemed calling gadgets merely. Various applications are used to browse the Internet, thus access financial data, and store sensitive personal information. In consequence, mobile devices are exposed to several types of attacks. Specifically, phishing attacks can easily take advantage of the limited or lack of security and defense applications therein. Furthermore, the limited power, storage, and processing capabilities render machine learning techniques inapt to classify phishing and spam emails in such devices. The present study proposes a distributed architecture hinging on machine learning approaches to detect phishing emails in a mobile environment based on a modified version of Bayesian Additive Regression Trees (BART). Apparently, BART suffers from high computational time and memory overhead, therefore, distributed algorithms are proposed to accommodate detection applications in resource constrained wireless environments.**

I. Introduction

Wireless and mobile technologies continue to prosper due to their convenience and portability. According to JiWire [14] there were more than 100,000 WiFi hotspots worldwide in 2006. Further, the total revenue of WLAN equipment is estimated to be $4.3 billion in 2009 as revealed by the Dell'Oro Group [10]. Moreover, users are using blackberries, personal digital assistants (PDAs), or even cell phones to store sensitive information and access financial data. Despite their convenience and ease of use, these wireless devices suffer from several limitations due to their limited power capacity. Processing capabilities and storage capacities are limited. These limitations certainly affect security and privacy solutions built for such devices to protect users against attacks [18]. Solutions that are designed to cope with such limitations need be light-weight, have less processing requirements, consume less storage, and thus require less power.

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 December 2007 Gartner Group published results of a survey showing that in 2007 phishing attacks in the U.S. increased compared to the past two years. In 2006, approximately, 3.25 million victims were spoofed by phishing attacks. In 2007, the number increased almost by 1.3 million victims. Moreover, in 2007, monetary losses, related to phishing, were estimated by $3.2 billion.

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 [1]. Moreover, several solutions proposed and implemented for detection and prevention of phishing attacks suffer from unacceptable levels of false positives or miss detection. Furthermore, most of these algorithms are based on server-side, computationally intensive, algorithms which will not lend easily to implementation on resource constrained mobile devices.

Previous research [2] showed that Bayesian Additive Regression Trees (BART) is a promising technique for spam classification. A modified version of BART, hereafter CBART, outperformed six other classifiers and achieved the maximum predictive accuracy on three different spam corpora. However, CBART performed the worst in computational time and required memory size. In order to overcome this latter drawback, we propose a distributed architecture for phishing detection. The contribution of this study is to propose a distributed client-server architecture to conceal the overhead caused by CBART, albeit take advantage of its high predictive accuracy. Note that our approach is not regarded as distributed classification nor distributed data mining. It is rather a distributed client-server architecture that exploits the competitive predictive accuracy of CBART and feeds it to other classifiers.

The rest of the paper is organized as follows. In Section II we discuss related work and the machine learning approaches demonstrated in the study. Section III introduces BART and the new modified model CBART. In Section IV we introduce the distributed approach for phishing detection. Section V demonstrates the experimental studies. We discuss the results in Section VI then conclude and motivate for future work in Section VII.

II. Related Work

Chandrasekaran et al. [7] proposed an approach to classify phishing based on phishing emails' structural properties. 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, were used in the study. 200 emails (100 phishing and 100 legitimate) were tested. Simulated annealing was applied as an algorithm for feature selection. After a feature set was chosen, information gain (IG) was used to rank these features based on their relevance. Thus, they applied one-class SVM to classify phishing emails based on the selected features. The results demonstrated a detection rate of 95% of phishing emails with a low false positive rate.

Fette et al. [11] 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. The authors claim that the methods can be used in the detection of phishing websites as well. 860 phishing emails and 6950 legitimate emails were tested. The proposed method detected correctly 96% of the phishing emails with a false positive rate of 0.1%. Ten handpicked features were selected for training using a phishing dataset that 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. [1] 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 used a *bag-of-words* as their feature set and the results demonstrated that using a spam detection mechanism merely, i.e. bag-of-words only, achieves high predictive accuracy. However, obviously relying on textual features, results in high false positive rates, as phishing emails are very similar to legitimate ones. 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.

To our best knowledge, there exist no research studies that investigate the application of phishing detection via machine learning in a mobile environment. Current studies and solutions either scrutinize the effectiveness of existing machine learning techniques in the phishing domain, or focus on building applications to detect phishing in mobile devices. To the contrary, our approach utilizes the competitive performance of machine learning techniques and applies distributed architectures to conceal the overhead associated with them.

We note that 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). 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 briefly describe the classifiers used in our experiments.

A. 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. 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 [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.

B. 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.

C. 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 [15]. 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 plays more role during the testing phase. Figure 1 depicts an example of neural network. The neural network in the

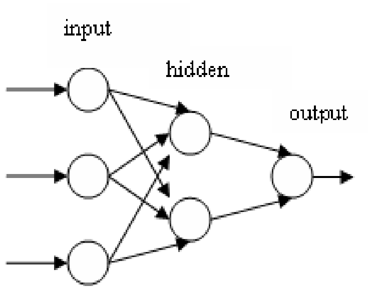

Fig. 1. Neural Network.

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 [16]

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

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.

D. 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.

E. 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 [15].

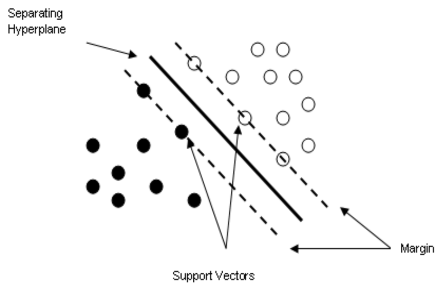

Fig. 2. Support Vector Machines.

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.

III. Introduction to BART

Bayesian Additive Regression Trees (BART) is a new learning technique, proposed by Chipman et al. [9], 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, we describe how to modify the current BART model and make it applicable to classification problems in general and phishing (or spam) classification in particular. 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 [12]. Here, we should mention that an algorithm that modifies and uses BART for binary classification, written in the statistical package $\mathbf{R}$, is provided online by the original authors of BART [9]. However, we developed our own algorithm in this work simultaneously, due to the fact that their algorithm was not available at the time we started this work. Also, to our best knowledge, no technical details or applications were provided by them for using BART for classification.

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 3, 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.

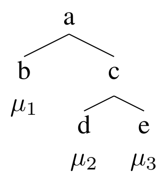

Fig. 3. A binary tree structure

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 $i = 1, \cdots, m$. Let M_i be the vector containing all terminal node parameters of T_i such that $M = \{M_1, \cdots, M_{b_i}\}$ 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 4 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 4).

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). In what follows, we describe the Bayesian implementation of BART very briefly. It is by no means self-contained. For a complete illustration, see [9] and the references therein.

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 $j = 1, \cdot, m$, where j is the total number of trees available.
- Sample σ given Y and all Ts, all Ms and σ.

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 [12]. 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.

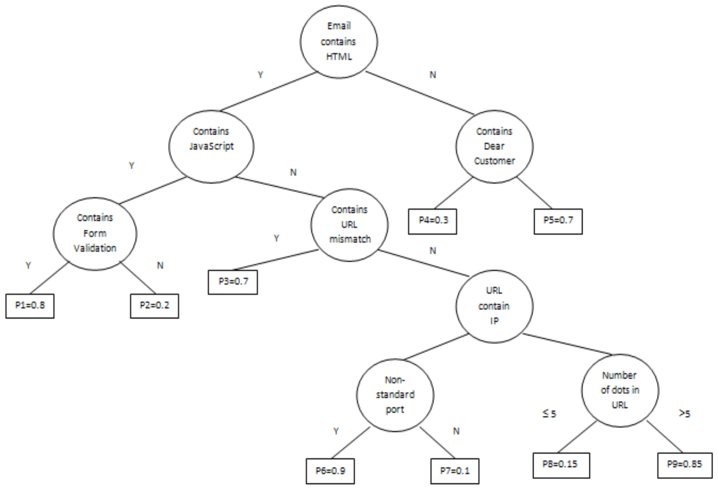

Fig. 4. Example of a binary tree.

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.

A. BART for Classification (CBART)

As mentioned in Section III, 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 [3], 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 [9] 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 our 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. Our 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 $(T_{(j)}, M_{(j)}, Z)$:

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

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 $\sigma = 1$) 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. There is no difficulty to implement our algorithm, especially because the original one has been provided as free open source software.

We now briefly 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.

IV. Distributed Phishing Detection

CBART suffers from high overhead in computation time and memory usage when compared to other classifiers. Generally, this is due to MCMC simulations required to draw the posterior probabilities and is regarded as a known drawback of the Bayesian approach [2]. Therefore, the implementation of CBART is impractical in resource constrained devices due to several limitations, albeit suitable in servers due to the abundance of resources (processing, power, and memory). Yet, we can take advantage of the superior predictive accuracy of CBART to improve the predictive accuracy in client devices. The basic idea is to use the predicted output by CBART and feed it to resource constrained clients in order to improve their predictive accuracy.

In the client side, a light weight classifier is needed to accommodate the limitations in client devices. Two vital characteristics need exist in such classifier; low computation time and memory overhead and competitive predictive accuracy. Based on the results in [2], CART requires the least amount of memory and takes the minimum computational time to predict spam emails. In addition, the predictive accuracy and the area under the curve (AUC) of CART are comparable to, yet do not outperform, other classifiers, hence the predictive accuracy of CART needs to be improved. As we mentioned earlier, we expect that this improvement can be accomplished by feeding the predicted output of CBART to the clients and adding it as a new feature to the dataset.

In Figure 5 we depict a block diagram of the distributed architecture. First, CBART is trained on a subset of the phishing dataset, thus used to predict the status of the testing set. Secondly, the predicted output of CBART is fed to the clients and added as a new feature to the testing subset. Now, the client devices are introduced to new data and CART is used to predict the status of new emails.

V. Experiential Studies

A. Phishing Dataset

6561 raw emails are used in building the dataset, from which 1409 emails are phishing. These emails are donated by [17] covering many of the new trends in phishing and collected between August 7, 2006 and August 7, 2007. The total number of legitimate emails is 5152, which are collected from 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. Table I shows that the percentage of these emails is 3% of the complete dataset. The remaining 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.

TABLE I
Corpus description.

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

The dataset constitutes of 71 features, in which the first feature represent the class of the email, whether it is phishing =1 or legitimate =0. Thus, the following 60 features represent the terms that frequently appear in phishing emails gauged by term frequency inverse document frequency (TF/ IDF). 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. Therefore, terms that appear often in a document and do not appear in many documents have a higher weight [4]. The last 10 features represent structural characteristics of phishing emails and several styles used by phishers to lure victims to make phishing emails look legitimate.

B. Experimental Setup

The area under the receiver operating characteristic (ROC) curve (AUC) is used as the primary measure to compare the performance of classifiers. In [13] the authors prove theoretically and empirically that AUC is more accurate than error rate to evaluate classifiers' performance. The AUC shows the trade off between the false positives and true positives at different cut-off points. Although classifiers' error rate (W_{Err}) or sometimes classifiers' accuracy (W_{Acc}) have been widely used in comparing classifiers' performance, they have

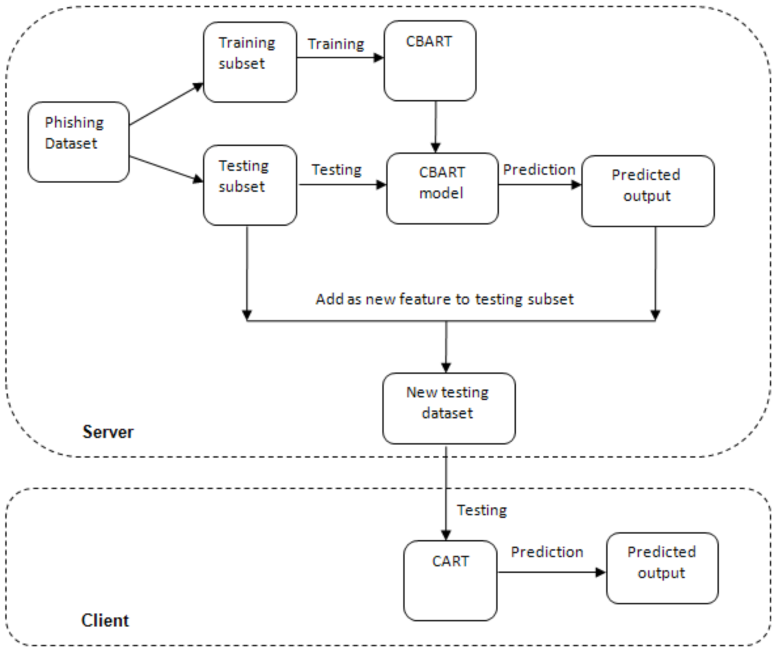

Fig. 5. Distributed phishing detection using feature addition block diagram.

been criticized for highly depending on the probability of the threshold chosen to approximate the positive classes. Here we note that, when using the error rate, we assign new classes to the positive class if the probability of the class is greater than or equal to 0.5 (i.e. threshold=0.5).

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:

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

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

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

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

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

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

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

In order to stay consistent with previous research though, we also compare the error rate of classifiers. According to [19] and [20] the predictive accuracy of classifiers is measured by the *weighted error* (W_{Err}). We assign equal weights on legitimate and phishing emails, hence $\lambda = 1$. Now, the weighted error rate $W_{Err}(\lambda)$, can be calculated as follows

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

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 in the study can perform automatic variable selection. To be fair, we use L1-SVM and penalized LR, where variable selection is performed automatically. The optimum classifiers' parameters are summarized in Table II.

TABLE II
OPTIMIZED INPUT PARAMETERS IN CLASSIFIERS.

Classifier	Input parameters	
CBART	number of trees= 100	$power = 1$
LR	$\lambda = 1 \times 10^{-4}$	
RF	number of trees= 50	
SVM	$\gamma = 0.1$	cost $(c) = 12$
NNet	size $(s) = 35$	weight decay $(w) = 0.7$

C. Experimental Results

In this section we present the experimental results by measuring the AUC using the complete dataset. In addition, we compare the FP, FN, and W_{Err} measures using the optimum parameters achieved from the previous section.

In Table III, we compare the AUC before and after applying the distributed approach on the complete dataset. Figure 6 and Figure 7 depict the ROCs for all classifiers before and after using the distributed approach respectively. Furthermore, Table IV and Table V compare the error rate, false positive, and false negative rates before and after applying the distributed approach respectively.

TABLE III
COMPARISON OF AUC BEFORE AND AFTER APPLYING THE DISTRIBUTED APPROACH. THE HIGHER THE AUC, THE BETTER THE CLASSIFIER'S PERFORMANCE.

Classifier	AUC before	AUC after	Increase/ decrease in AUC
CART	96.06%	97.55%	+1.49%
LR	54.45%	51.45%	-3.0%
RF	95.48%	95.65%	+0.17%
SVM	97.18%	97.24%	+0.06%
NNet	98.80%	98.84%	+0.04%

TABLE IV
ERROR RATE, FALSE POSITIVE, AND FALSE NEGATIVE RATES BEFORE APPLYING THE DISTRIBUTED APPROACH.

Classifier	W_{Err}	FP	FN
CART	7.00%	11.55%	22.10%
RF	3.68%	4.25%	13.20%
SVM	2.39%	5.43%	13.77%
LR	5.34%	7.29%	18.38%
NNet	4.31%	6.16%	14.32%

TABLE V
ERROR RATE, FALSE POSITIVE, AND FALSE NEGATIVE RATES AFTER APPLYING THE DISTRIBUTED APPROACH.

Classifier	W_{Err}	FP	FN
CART	2.97%	3.01%	11.08%
RF	2.85%	2.60%	10.90%
SVM	3.07%	3.09%	11.45%
LR	3.37%	4.14%	11.83%
NNet	3.27%	3.77%	11.74%

TABLE VI
INCREASE OR DECREASE IN ERROR RATE, FALSE POSITIVE, AND FALSE NEGATIVE RATES AFTER APPLYING THE DISTRIBUTED APPROACH. THE LOWER THE ERROR RATE, FALSE POSITIVE, AND FALSE NEGATIVE RATES, THE BETTER THE CLASSIFIER'S PERFORMANCE.

Classifier	W_{Err}	FP	FN
CART	-4.03%	-8.54%	-11.02%
RF	-0.83%	-1.65%	-2.30%
SVM	0.68%	-2.34%	-2.32%
LR	-1.97%	-3.15%	-6.55%
NNet	-1.04%	-2.39%	-2.58%

VI. DISCUSSION

The present study investigates detecting phishing emails using a distributed architecture. A client-server architecture

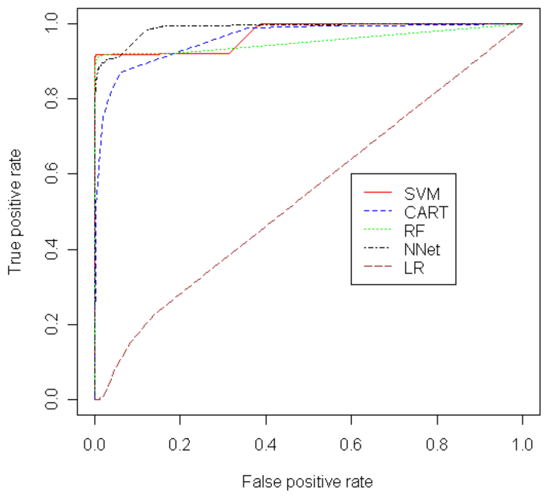

Fig. 6. ROC for all classifiers using the complete dataset before applying the distributed approach.

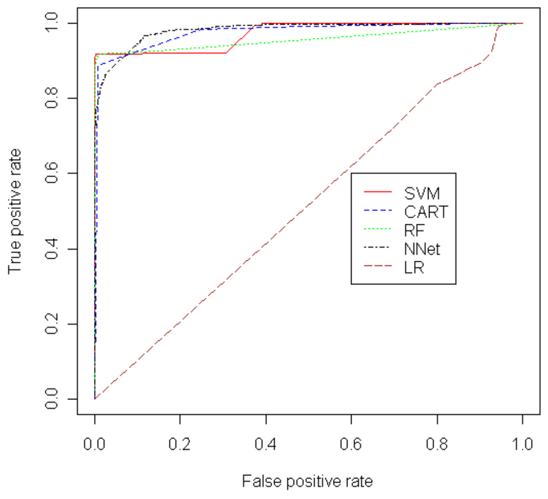

Fig. 7. ROC for all classifiers using the complete dataset after applying the distributed approach.

is applied to exploit the superior detection performance of CBART and correspondingly conceal the computational overhead and memory requirement associated with it. The results demonstrate that the performance of potential classifiers at the clients, namely CART, SVM, NNet, and RF improves after using the predicted output of CBART in their datasets. CART achieves the maximum improvement in AUC of 1.49%. Despite the improvement in other classifiers, namely, RF by 0.17%, SVM by 0.06%, and NNet by 0.04%, apparently, the improvement in the AUC is unnoticeable. Unlike other classifiers, the performance of LR worsens with a performance decay of 3.0%. Figure 8 depicts the performance improvement or decay for each of the classifiers separately.

The results show (see Table VI) that the predictive accuracy of CART improves by 57.52%, leaving behind all rivals. In addition, the predictive accuracy of LR improves by 36.95%, followed by NNet with 24.13%, then RF with 22.44%. Strangely, the predictive accuracy of SVM decreases by 28.33%.

Similarly, the FP rate of CART decreases by 73.97%, followed by LR with 43.26%, SVM with 43.12%, RF with 38.71%, and lastly NNet with 38.86% decrease. The FN rate of CART decreases by 49.89%, followed by LR with 35.63%, NNet with 18.04%, RF with 17.44%, and lastly SVM with 16.87% decrease.

Clearly, the results show that CART outperforms all rivals in terms of performance improvement. Add to that, the low computational overhead and memory requirement associated with CART demonstrated in previous research [2]. In consequence, a long with CBART, we expect CART to be a suitable candidate for phishing detection in the proposed distributed architecture.

VII. Conclusions and Future Work

There has been an aggregative rise in phishing attacks in the past couple of years; however, there seems to be no solution to subvert such threats. Recently, mobile devices, be it cell phones, PDAs, or others have been frequently used to store sensitive information and access financial accounts. Apparently, the lack of security applications, limited processing and storage capabilities, and power constraints, render such devices prone to new vectors of phishing attacks. The present study proposed a distributed architecture to detect phishing attacks in a mobile environment. CBART was implemented in a central server and associated resource constrained clients used CART, as it showed to be lighter in computational time and memory overhead and competitive in predictive accuracy as well.

Empirically, the results demonstrated that the superior predictive accuracy of CBART can be exploited to enhance the performance of other classifiers. Feeding the predicted output of CBART to light weight classifiers deployed in resource constrained devices proved to improve their performance. After applying the proposed distributed architecture, detection applications in resource constrained devices, namely CART, achieved the maximum AUC, error rate, false positive rate, and false negative rate improvements compared to all other rivals. CART achieved 1.49% AUC improvement and 57.52% decrease in the error rate. The FP rate of CART improved by 73.97% and the FN rate improved by 49.89%.

The results motivate future work to explore the variable selection feature in CBART and compare it to other well-known variable selection approaches. Further, the automatic variable selection feature in CBART can be scrutinized to examine the improvement in classifiers' performance in the aforementioned distributed architecture.

References

[1] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair. A comparison of machine learning techniques for phishing detection. In *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 60–69, New York, NY, USA, 2007. ACM.

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

[3] J. H. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993.

[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, 2006. Available from: http://faculty.chicagogsb.edu/robert.mcculloch/research/code/BART-7-05.pdf.

[10] Dell'Oro Group. Wireless LAN market to double [online]. 2005. Available from: http://www.jiwire.com/press-100k-hotspots.htm [cited 26 April 2008].

[11] 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.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

[13] J. Huang and C. X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3), 2005.

[14] JiWire. Worldwide Wi-Fi hotspots hits the 100,000 mark [online]. 2006. Available from: http://www.jiwire.com/press-100k-hotspots.htm [cited 26 April 2008].

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

[16] 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.

[17] J. Nazario. Phishing Corpus [online]. 2007. Available from: http://monkey.org/~jose/phishing/phishing3.mbox [cited 26 April 2008].

[18] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady. Security in embedded systems: Design challenges. *Trans. on Embedded Computing Sys.*, 3(3):461–491, 2004.

[19] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6(1):49–73, 2003.

[20] 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.

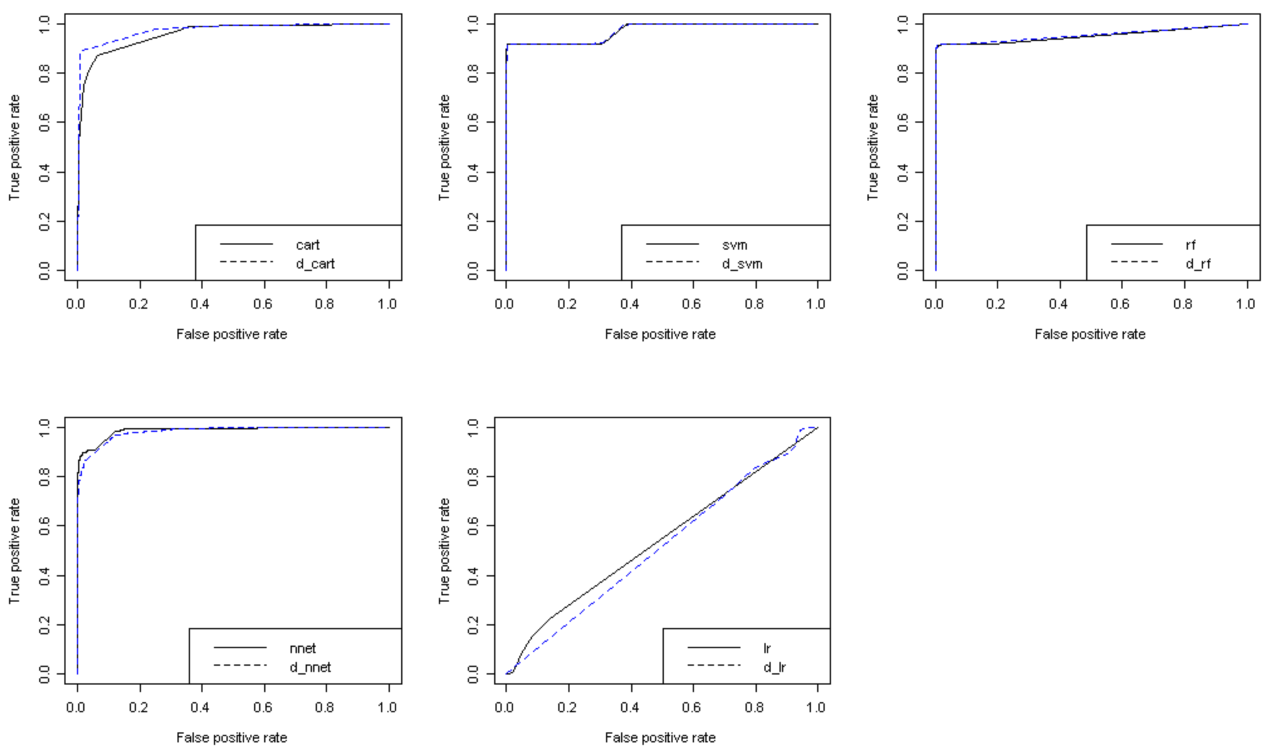

Fig. 8. Comparison of the ROCs for individual classifiers before and after applying the distributed approach. The solid line depicts the ROC before the distributed approach and the dashed line depicts the ROC after the distributed approach.