# An Evaluation of Machine Learning-based Methods for Detection of Phishing Sites

*Daisuke Miyamoto*[*], *Hiroaki Hazeyama*[*], *Youki Kadobayashi*[*]

*Nara Institute of Science and Technology*
*Graduate School of Information Science*
*Internet Engineering Laboratory*
*8916-5 Takayama, Ikoma, Nara, Japan*

## Abstract

*In this paper, we present the performance of machine learning-based methods for detection of phishing sites. We employ 9 machine learning techniques including AdaBoost, Bagging, Support Vector Machines, Classification and Regression Trees, Logistic Regression, Random Forests, Neural Networks, Naive Bayes, and Bayesian Additive Regression Trees. We let these machine learning techniques combine heuristics, and also let machine learning-based detection methods distinguish phishing sites from others. We analyze our dataset, which is composed of 1,500 phishing sites and the same number of legitimate sites. We then classify them using the machine learning-based detection methods, and measure the performance. In our evaluation, we used $f_1$ measure, error rate, and Area Under the ROC Curve (AUC) as performance metrics along with our requirements for detection methods. The highest $f_1$ measure is 0.8581, the lowest error rate is 14.15%, and the highest AUC is 0.9342, all of which are observed in the case of AdaBoost. We also observe that 7 out of 9 machine learning-based detection methods outperform the traditional detection method.*

**Keywords: phishing, web spoofing, AdaBoost, Bagging, SVM, CART, Logistic Regression, Random Forests, Neural Networks, Naive Bayes, BART**

## 1 Introduction

Phishing is a form of identity theft whose targets are users rather than computer systems. A phishing attacker attracts victims to a spoofed web site, a so-called "phishing site", and attempts to persuade them to send their personal information.

To prevent a user from browsing phishing sites, there are two distinct approaches. One is URL filtering. It detects phishing sites by comparing the URL of a site a user visits with a URL blacklist composed of the URLs of phishing sites. However, it is difficult to build a perfect blacklist due to the rapid increase of phishing sites. According to trend reports published by the Anti-Phishing Working Group [3], the number of reported phishing sites was 25,630 in March 2008, far surpassing the 14,315 in July 2005.

The other approach is a heuristic-based solution. A heuristic is an algorithm to distinguish phishing sites from others based on users' experience, that is, a heuristic checks if a site seems to be a phishing site. A heuristic-based solution employs several heuristics and converts results from each heuristic into a vector. Based on the vector, the heuristic-based solution calculates the likelihood of a site being a phishing site and compares the likelihood with the defined discrimination threshold. Different from URL filtering, a heuristic-based solution has a possibility to identify new phishing sites.

Unfortunately, the detection accuracy of existing heuristic-based solutions is far from suitable for practical use [23], even if various studies [14, 20, 21] discovered heuristics. To improve the detection accuracy, both discovering innovative heuristics and refining the calculation algorithm of the likelihood are important.

In our previous work [15], we attempted to employ a machine learning technique to improve the detection accuracy. We employed AdaBoost, a machine learning technique, as a calculation method of the likelihood. Our preliminary evaluation showed the AdaBoost-based detection method can achieve higher detection accuracy than the traditional detection method.

Here, we present a performance evaluation of 9 Machine Learning-based Detection Methods (MLBDMs) including AdaBoost, Bagging, Support Vector Machines (SVM), Classification and Regression Trees (CART), Logistic Regression (LR), Random Forests (RF), Neural Networks (NN), Naive Bayes (NB) and Bayesian Additive Regression Trees (BART). In the evaluation, we used $f_1$ measure, error rate, Area Under the ROC Curve (AUC) as performance metrics along with our requirements for detection methods. Our re-

quirements are (i) they must achieve high detection accuracy, (ii) they must adjust their detection strategies for web users.

We let all MLBDMs classify whether a site is a phishing site or not by using a dataset of 3,000 URLs, composed of 1,500 phishing sites reported during November, 2007 – February, 2008, and the same number of legitimate sites. We employ 8 heuristics presented in CANTINA [24] and measure their performance in a less biased way. The results show that the highest $f_1$ measure is 0.8581, the lowest error rate is 14.15%, and the highest AUC is 0.9342, all of which are observed in the case of AdaBoost.

The rest of this paper is organized as follows: In Section 2, we present our related work, and we introduce the machine learning techniques in Section 3. In Section 4, we describe our evaluation conditions, and we show our experimental results in Section 5. We discuss whether or not MLBDMs can continue to provide better performance in Section 6, and show our future work in Section 7. Finally, we summarize our contributions in Section 8.

## 2 Related Work

For mitigating phishing attacks, machine learning, which facilitates the development of algorithms or techniques by enabling computer systems to learn, has begun to garner attention. PFILTER, which was proposed by Fette et al. [11], employed SVM to distinguish phishing emails from other emails. According to [1], Abu-Nimeh et al. compared the predictive accuracy of several machine learning methods including LR, CART, RF, NB, SVM, and BART. They analyzed 1,117 phishing emails and 1,718 legitimate emails with 43 features for distinguishing phishing emails. Their research showed that the lowest error rate was 7.72% in the case of Random Forests. In [5], Ram Basnet et al. performed an evaluation of six different machine learning-based detection methods. They analyzed 973 phishing emails and 3,027 legitimate emails with 12 features, and showed that the lowest error rate was 2.01%. The experimental conditions were different between [1] and [5], however, the machine learning provided high accuracy for the detection of phishing emails.

Aside from phishing emails, a machine learning method was also used to detect phishing sites. According to [17], Pan et al. presented an SVM-based page classifier for detection of phishing sites. They analyzed 279 phishing sites and 100 legitimate sites with 8 features, and the results showed that the average error rate was 16%.

Our previous work [15] employed AdaBoost for the detection of phishing sites. We checked 100 phishing sites and the same number of legitimate sites with 7 heuristics. We let AdaBoost to assign weights on heuristics by training with 50 legitimate sites and the same number of phishing sites, and tested the detection accuracy by using the rest of sites. Our performance evaluation showed that the average error rate was 14.7%.

We find that there are two problems in earlier research. One is that the number of features for detecting phishing sites is lesser than that for detecting phishing emails. It indicates that the detection of phishing sites is much difficult than that of phishing emails. The other is that no research contribution confirmed whether any kind of MLBDMs were available to distinguish phishing sites from legitimate sites. To the best of our knowledge, earlier research tested only one machine learning technique. In this paper, we evaluate 9 MLBDMs and show their performance.

## 3 Overview of Machine Learning Techniques

In this section, we briefly explain each machine learning technique which is used in our evaluation.

### 3.1 AdaBoost

Adaptive Boosting (AdaBoost) [13] algorithm learns a "strong" algorithm by combining a set of "weak" algorithms $h_t$ and a set of weight $\alpha_t$:

$$H_{Ada} = \sum \alpha_t * h_t. \tag{1}$$

The weights are learned through supervised training off-line [12]. Formally, AdaBoost uses a set of input data $\{x_i, y_i : i = 1, \ldots, m\}$ where $x_i$ is the input and $y_i$ is the classification.

Each weak algorithm is only required to make the correct detections slightly over half the time. The AdaBoost algorithm iterates the calculation of a set of weight $D_t(i)$ on the samples. At $t = 1$, the samples are equally weighted so $D_1(i) = 1/m$.

The update rule consists of three stages. First, AdaBoost chooses the weight as shown in Equation 2.

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{2}$$

where $\epsilon_t = Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$. Second, AdaBoost updates the weight by Equation 3.

$$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if \quad h_t(x_i) = y_i \\ e^{\alpha_t} & if \quad h_t(x_i) \neq y_i \end{cases} \tag{3}$$

where $Z_t$ is a normalization factor, $\sum_{i=i}^{m} D_{t+1}(i) = 1$.

### 3.2 Bagging

Bootstrap Aggregation (Bagging) [6] is a typical ensemble learning method, and its key feature is that dataset is perturbed by resampling with replacement. Given $n$ samples in dataset, bagging selects $m$ ($m$ <n) samples for training and constructs a classifier $h$. Taking $B$ iterations, it outputs the final classifier by majority vote of $h_1, h_2, \ldots h_B$ as shown in Equation 4.

$$f_{bagging} = \text{argmax}_y \sum_{i=1}^{B} (h_i = y) \tag{4}$$

### 3.3 Support Vector Machines

Support Vector Machines (SVM) [10] is also one of the typical machine learning methods for classification and regression. The key idea of SVM is to map data from the input space into a higher dimensional feature space, and to find the optimal separating hyperplane between two classes by maximizing the margin between the classes' closest points.

A discriminating hyperplane will satisfy Equation 5.

$$\begin{array}{lll} w'x_i + w_0 \geq 0 & if & t_i = +1 \\ w'x_i + w_0 < 0 & if & t_i = -1 \end{array} \tag{5}$$

where the distance of any point $x$ to a hyperplane is $|w'x_i + w_0|/||w||$ and the distance to the origin is $|w_0|/||w||$.

### 3.4 Logistic Regression

Logistic Regression (LR) [4] is a model used for binary data prediction. LR is designed to deal with confounding variables, and its model typically uses the *logit* function as shown in Equation 6.

$$log \frac{p(x)}{1 - p(x)} = \beta_0 + \sum \beta_i x_i \tag{6}$$

where $x$ is a vector of predictors and $\beta$ is a vector of regression parameters.

### 3.5 Classification and Regression Trees

Classification and Regression Trees (CART) [8] is a typical decision tree algorithm. The modeling and prediction within the CART analysis is accomplished through a recursive binary partitioning of a training dataset. Each parent node is split into two child nodes and the procedure of splitting is repeated by treating each child node as a parent node. When a data node cannot be split into additional child nodes, it is called a terminal node. Once the first terminal node has been created, the algorithm repeats the procedure for each set of data until all data are categorized as terminal nodes.

CART requires a measure of node impurity and generally employs Gini Index as an impurity function. In a node $t$, the Gini Index criterion assigns a sample to a class $c_i$ with the probability $p(c_i|t)$. The estimated probability of misclassification under this rule is as shown in Equation 7.

$$\text{Gini Index} = 1 - \sum (p(c_i|t))^2 \tag{7}$$

### 3.6 Random Forest

Random Forest (RF) [7] is a classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. In building each decision tree model based on a different random subset of the training dataset, a random subset of the available variables is used to choose how best to partition the dataset at each node. Each decision tree is built to its maximum size, with no pruning performed.

The basic idea is similar to Bagging. The main difference between Bagging and RF is that RF uses a random subset of the available variables whereas Bagging uses all available variables. So, RF is suitable for handling a very large number input variables.

### 3.7 Neural Network

Neural Network (NN) is a non-linear and parallel computation model which is referred to a network of biological neurons. NN has overwhelming strengths in learning ability, auto-adapting ability, generalization performance and anti-noise ability.

The neurons are organized into three types of layers. The input layer presents the feature vector of input variables. The next layer is called a hidden layer; NN assumes that there may be several hidden layers. The final layer is the output layer, where there is one node for classification. Since interconnections do not loop back or skip other neurons, the network is called *feedforward*.

### 3.8 Naive Bayes classifier

The Naive Bayes (NB) classifier is a simple but effective classifier that has been used in numerous applications such as email filtering. Generally, NB's computational time is less than the non-naive Bayes approach because NB is based on Bayes' theorem with the independent feature model. The model of prediction was formulated in Equation 8.

$$f_{nb} = \text{argmax}_y P(y) \Pi(x_i|y) \tag{8}$$

### 3.9 Bayesian Additive Regression Trees

Bayesian Additive Regression Trees (BART) [9] is designed for discovering unknown function $f$ that predicts an output $Y$ using a $p$ dimensional vector of inputs

$x$. The basic idea of BART is to model by a sum of regression trees,

$$f(x) = \sum g(x) \tag{9}$$

where function $g$ denotes a binary regression tree. Replacing $f$ in Equation 9 by modeling or approximating $f(x)$, BART obtains Equation 10 where $\epsilon$ is the random error.

$$Y = \sum g(x) + \epsilon, \epsilon \sim N(0, \sigma^2). \tag{10}$$

Conceptually, BART can be viewed as a Bayesian nonparametric approach which fits a parameter rich model using a strongly influential prior distribution. BART does not require variable selection, which is performed automatically as the trees are built. In addition, in order to fit the sum-of-trees model, BART uses a tailored version of Bayesian backfitting Markov Chain Monte Carlo simulation that iteratively constructs and fits successive residuals.

# 4 Evaluation Approach

In this paper, we evaluate the performance of MLBDMs. We let each machine learning method combine heuristics, perform supervised learning from the dataset, and distinguish phishing sites from other sites.

In this section, we define metrics of the performance evaluation along with our requirements for MLBDMs. We then decide the heuristics that we used in our evaluation, and describe how we construct a dataset for both training and testing. Finally, we explain the preliminary set-up of our experiments.

## 4.1 Evaluation Metrics

First, we defined metrics for evaluating performance along with requirements for detection methods. Our requirements were as follows.

1. *Accuracy*
   An MLBDM must achieve high detection accuracy. User safety would obviously be compromised if phishing prevention systems labeled phishing sites as legitimate. Users would also complain if prevention systems labeled legitimate sites as phishing sites because of the interruption in browsing caused by prevention systems.

2. *Adjustment Capability*
   An MLBDM must adjust its strategy for detecting phishing sites for web users. If a user is a novice, who is easily taken in by phishing attacks, phishing prevention systems should decrease false negative errors instead of increasing false positive errors. Conversely, if a user is a security expert, the system should focus on decreasing false positive errors.

| | actual phishing sites | actual legitimate sites |
|---|---|---|
| predict phishing sites | $tp$ | $fp$ |
| predict legitimate sites | $fn$ | $tn$ |

**Table I. Test Result**

For Requirement 1, we used the $f_1$ measure (higher is better) and the error rate (lower is better) as metrics to evaluate the detection accuracy. Statistically, $f_1$ measure has been used as an index of a test's accuracy. This measure is the harmonic mean of precision and recall. Given the test result as shown in Table I, precision $p$ equals $tp/(tp + fp)$ and recall $r$ equals $tp/(tp + fn)$. The $f_1$ measure can be calculated by $2 \cdot p \cdot r/(p + r)$. The average error rate has been also a reasonable metric to indicate the detection accuracy. It is calculated by dividing the number of incorrectly identified sites by the number of all sites in the dataset. So, the error rate equals $(fp + fn)/(tp + tn + fp + fn)$.

For Requirement 2, we performed Receiver Operating Characteristic (ROC) analysis. Generally, detection methods calculate the likelihood of being phishing sites $L$ and compare the likelihood with the defined discrimination threshold $\theta$. In our experiment, MLBDMs distinguish a phishing site by checking if $L$ is less or equal than $\theta(= 0)$. Imagine that $\theta$ was higher than 0. In this case, MLBDMs would tend to label a site as phishing rather than as legitimate. Conversely, MLBDMs would tend to label a site as legitimate if $\theta$ was lower than 0. Accordingly, we assumed that adjusting $\theta$ provides different detection strategies. Based on this assumption, we employed ROC analysis because it has been widely used in data analysis to study the effect of varying the threshold on the numerical outcome of a diagnostic test. We also used the Area Under the ROC Curve (AUC; higher is better) as a metric to evaluate adjustment capability.

## 4.2 Heuristics

In our evaluation, we employ 8 heuristics as follows.

- Age of Domain
  Checking if the domain was registered more than 12 months ago. If the site has been registered more than 12 months, the heuristic deems it a legitimate site, and otherwise it deems it a phishing site.

- Known Images
  Checking if a page contains inconsistent well-known logos such as eBay, PayPal, Citibank, Bank of America, Fifth Third Bank, Barclays Bank, ANZ Bank, Chase Bank, and Wells Fargo Bank. For example, if a site contains the eBay logos but is not on an eBay domain, the heuristic deems this site a phishing site.

- Suspicious URL
  Checking if a URL of the site contains an "at" symbol (@) or a "dash" (-) in the domain name. If so, the heuristics deems it a phishing site because phishing attackers are likely to use these symbols in the domain name of a phishing site. When the at "@" symbol is used in a URL, all text before the @ symbol is ignored and the browser references only the information following the @ symbol as a hostname. Phishing attackers likely abuse this URL scheme: For example, if `http://paypal.com@phishing.com` is used, web browsers would be directed to the `phishing.com`. Even if it seemed like `paypal.com`, web browsers would ignore this.

- Suspicious Links
  Similar to the Suspicious URL heuristic, this one checks if a link on the page contains an "at" symbol or a dash.

- IP Address
  Checking if the domain name of the site is an IP Address. Although legitimate sites rarely link to pages by an IP address, phishers often attract victims to phishing sites by IP address links.

- Dots in URL
  Checking if the URL of the site contains five or more dots. According to [11], dots can be abused for attackers to construct legitimate-looking URLs. One technique is to have a sub domain. Another is to use a redirection script, which to the user may appear like a site hosted at google.com, but in reality will redirect the browser to phishing.com. In both of these examples, either by the inclusion of a URL into an open redirect script or by the use of a number of sub domains, there are a large number of dots in the URL.

- Forms
  Checking if the page contains any web input forms. In the case of CANTINA, it scans the HTML for `<input>` tags that accept text and are accompanied by labels such as "credit card" and "password." If so, the heuristic deems it a phishing site.

- TF-IDF-Final
  This heuristic checks if the site is phishing by employing TF-IDF-Final, which is an extension of the Robust Hyperlinks algorithm [18]. When the heuristic attempts to identify phishing sites, it feeds the mixture word lexical signatures and a domain name of the current web site into Google. If the domain name matches the domain name of the top 30 search results, the web site is labeled legitimate.

These 8 heuristics were employed in CANTINA [24]. To the best of our knowledge, the most successful tool for combining heuristics is CANTINA, which has achieved high accuracy of detecting phishing sites without using the URL blacklist.

In CANTINA, each heuristic returns 1 binomial variable. Based on the result of each heuristic, CANTINA calculates the likelihood of being a phishing site ($L$) by weighted majority as shown in Equation 11.

$$L = \sum W_i * h_i \tag{11}$$

A positive value for $L$ means that it is labeled as legitimate, while a negative value or zero means that it is labeled as a phishing site. Zhang et al. mentioned that a heuristic should have high accuracy in detecting phishing sites while also having a low false positive rate. Thus, they assigned weight by calculating the true positive rate minus the false positive rate. Given the effect $e_i$ of each heuristic, they calculated each weight proportionally, that is:

$$W_i = \frac{e_i}{\sum e_i} \tag{12}$$

## 4.3 Dataset

We then built a dataset with the criteria for choosing URLs. Based on the criteria in the original CANTINA, we collected URLs with the same number of phishing sites and legitimate sites. All sites were English language sites because CANTINA does not work well if the sites are not written in English. First, we chose 1,500 phishing sites that were reported on PhishTank.com [16] from November, 2007 to February, 2008. Second, we also selected 227 URLs from 3Sharp's study of anti-phishing toolbars [19]. There were listed 500 URLs of legitimate sites in [19], however, we could not connect to many listed URLs. Third, we attempted to collect 500 URLs from Alexa Web Search [2] and observed 477 URLs. Finally, we gathered 796 URLs from yahoo random link [22].

Each site was checked with our implementation of heuristics, and was converted into a vector $\vec{x} = (x_1, x_2 \dots x_p)$, where $x_1 \dots x_p$ are the values corresponding to a specific feature. The dataset consisted of 8 binary explanatory variables and 1 binary response variable.

To perform our evaluation in a less biased way, we employed 4-fold cross validation. Furthermore, our cross validation was repeated 10 times in order to average the result.

## 4.4 Experimental Set-up

We adjusted the parameters for MLBDMs to minimize the error rate in training. For decision tree-based machine learning techniques such as RF, we tested

**Table II. Precision, Recall and $f_1$ measure, False Positive Rate(FPR), False Negative Rate(FNR), Error Rate(ER), and AUC**

| | *Precision* | *Recall* | $f_1$ *measure* | FPR | FNR | ER | AUC |
|---|---|---|---|---|---|---|---|
| AdaBoost | 0.8614 | 0.8551 | 0.8581 | 13.83% | 14.49% | 14.15% | 0.9342 |
| Bagging | 0.8492 | 0.8573 | 0.8527 | 15.36% | 14.27% | 14.82% | 0.9231 |
| SVM | 0.8629 | 0.8498 | 0.8562 | 13.57% | 15.02% | 14.29% | 0.8926 |
| CART | 0.8330 | 0.8542 | 0.8384 | 18.16% | 14.58% | 16.37% | 0.9062 |
| LR | 0.8510 | 0.8588 | 0.8548 | 15.10% | 14.12% | 14.60% | 0.9172 |
| RF | 0.8566 | 0.8546 | 0.8554 | 14.37% | 14.54% | 14.45% | 0.9296 |
| NN | 0.8633 | 0.8512 | 0.8570 | 13.54% | 14.88% | 14.21% | 0.9310 |
| NB | 0.8464 | 0.8636 | 0.8547 | 15.74% | 13.64% | 14.69% | 0.9215 |
| BART | 0.8567 | 0.8550 | 0.8555 | 14.39% | 14.50% | 14.45% | 0.9321 |
| CANTINA | 0.9134 | 0.6519 | 0.7606 | 06.21% | 34.81% | 20.52% | 0.9162 |

them using different numbers of trees, namely 100, 200, 300, 400, and 500 trees. The minimum error rate (14.27%) was observed when the number of trees was 300, followed by 200 and 400 (14.28%), 500 (14.30%), and 100 (14.37%). Thus, we set the number of trees to 300 for RF-based detection methods.

The iteration time was set to 500 in all of our experiments if the machine learning technique needed to analyze iteratively for reducing training errors. The minimum error rate (14.27%) was observed when the number of iterations was 500, followed by 300 and 400 (14.28%), 200 (14.30%), and 100 (14.31%). In addition, finding the optimal iteration number is important, however, the choice of the exact value of the optimal iteration number is not often a critical issue since the increase in test error is relatively slow.

For some types of machine learning techniques, we used threshold value to approximate the prediction output. For example, BART is designed for regression, not for classification. Therefore, BART gives quantitative value whereas we need an MLBDM to output binary value that indicates whether a site is a phishing site or not. In such cases, we employed threshold value and observed if the result of BART regression was greater than the threshold. We decided the threshold in the same fashion as the original CANTINA. In the case of CANTINA, the maximum likelihood of being a phishing site is -1 and that of being a legitimate site is 1; therefore, it employs the middle value 0 as the threshold value. In SVM, we tested both linear and non-linear kernel functions. The average error rate in training by using *Radial Based Function* (RBF), one of the typical non-linear kernel functions, was 14.18%, less than 21.02% of linear kernel. Thus, we used RBF in our experiments.

In NN, we selected the number of units in the hidden layer, namely 1, 2, 3, 4, and 5 units, for finding the minimum average error rate. The minimum error rate (14.14%) was observed when the number of units was 5, followed by 4 (14.23%), 2 (14.46%), 3 (15.48%), and 1 (16.03%).

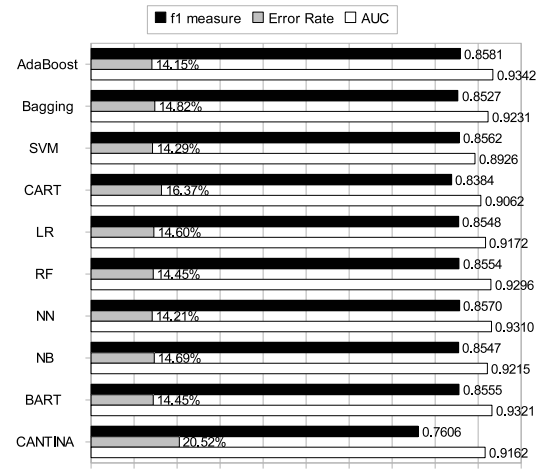


**Figure 1. Test Result of $f_1$ measure, Error Rate, and AUC**

## 5 Evaluation

In this section, we evaluate the performance of all MLBDMs by measuring $f_1$ measure, error rate and AUC, and studying them comparatively. We also compare MLBDMs with the original CANTINA.

First, we measured the accuracy of all MLBDMs. We calculated $f_1$ measure for each pattern of dataset respectively, and also calculated their average as shown in Table II. For readability, we summarized the performance of MLBDMs in Figure 1 where black bars denoted $f_1$ measure.

The highest $f_1$ measure was 0.8581 in AdaBoost, followed by NN (0.8570), SVM (0.8562), BART (0.8555),

RF (0.8554), LR (0.8548), NB (0.8547), Bagging (0.8527) and finally CART (0.8384). We observed that the highest precision was 0.8633 in NN and the lowest was 0.8330 in CART. The highest recall was 0.8636 in NB and the lowest was 0.8498 in SVM.

We then calculated the error rate in Figure 1, where gray bars denoted the error rate. The lowest error rate was 14.15% in AdaBoost, followed by NN (14.21%), SVM (14.29%), RF and BART (14.45%), LR (14.60%), NB (14.69%), Bagging (14.82%), and finally CART (16.37%). We observed that the lowest false positive rate was 13.54% in NN and the highest was 18.16% in CART. The lowest false negative rate was 13.64% in NB and the highest was 15.02% in SVM.

We also calculated AUC as shown in Figure 1, where white bars denoted AUC. The highest AUC was 0.9342 in AdaBoost, followed by BART (0.9321), NN (0.9310), RF (0.9296), Bagging (0.9231), NB (0.9215), LR (0.9172), CART (0.9062), and finally SVM (0.8956). Additionally, we plotted ROC curves of all MLBDMs as shown in Figure 2. For readability, each graph presented 3 ROC curves. We observed that all ROC curves passed through the upper left space in the graph. It indicated that all MLBDMs could achieve both high true positive rate and lower false positive rate because the best possible detection method would yield a point in the upper left corner (0,1) of the ROC space, representing that the true positive rate is 100% and the false positive rate is 0%.

Finally, we compared all MLBDMs with CANTINA's detection method. We evaluated the performance of CANTINA in the same way as that described in Section 4, and observed $f_1$ measure was 0.7607, error rate was 20.52%, and AUC was 0.9162 as shown in Figure 1, respectively. According to our comparison, 7 out of 9 MLBDMs, namely AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods, outperformed CANTINA.

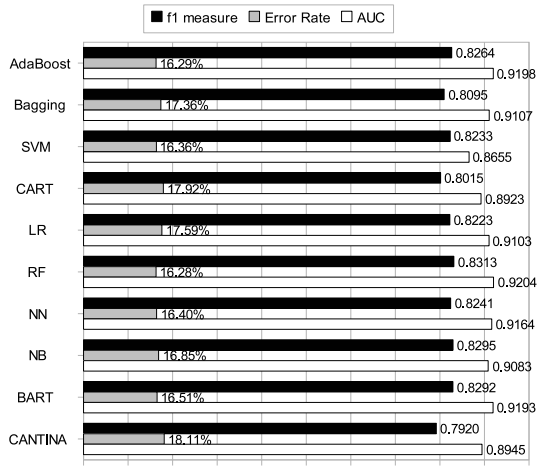


**Figure 3. Test Result of $f_1$ measure, Error Rate, and AUC by using new dataset**

## 6 Discussion

In this section, we discuss whether or not MLBDMs will continue to provide better performance. As we mentioned in Section 4.3, our collected phishing sites were reported on PhishTank.com from November, 2007 to February, 2008. Phishing attackers would attempt to build new phishing sites which are designed to evade detection, so we need to verify whether or not MLBDMs can keep higher performance in future.

For a preliminary analysis, we built new dataset which contained modern phishing sites. We collected 1,500 URLs of phishing sites reported on PhishTank.com from August, 2008 to October, 2008. We also gathered 1,500 URLs of legitimate sites in the same fashion as we described in Section 4.3. By using these 3,000 URLs, we adjusted the parameters for MLBDMs and measured their performance by doing 4-fold cross validation 10 times.

The results were shown in Figure 3 where black bars, gray bars, and white bars denoted the average $f_1$ measure, error rate, and AUC, respectively. The highest $f_1$ measure was 0.8313 in RF, followed by NB(0.8295), BART(0.8292), AdaBoost(0.8264), NN(0.8241), SVM(0.8233), LR(0.8223), Bagging(0.8095), and finally CART(0.8015). The lowest error rate was 16.28% in RF, followed by AdaBoost(16.29%), SVM(16.36%), NN(16.40%), BART(16.51%), NB(16.85%), Bagging(17.36%), LR(17.59%), and finally CART(17.92%). The highest AUC was 0.9204 in RF, followed by 0.9198(AdaBoost), BART(0.9193), NN(0.9164), Bagging(0.9107), LR(0.9103), NB(0.9083), CART(0.8923), and finally SVM(0.8655). Aside from MLBDMs, $f_1$ measure was 0.7920, error rate was 18.11%, and AUC was 0.8945 in the case of CANTINA. Similar to the comparison described in Section 5, 7 out of 9 MLBDMs, namely AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods, outperformed CANTINA. Accordingly, we predict that employing machine learning for detection of phishing sites has effectiveness.

## 7 Future Work

In our future work, we will implement a phishing prevention system according to the detection result. Within such a system, we should adjust the discrimination threshold for each web user, as we mentioned in Section 4.1.

Table III shows the false positive rate when the false negative rate was less than 5.00%, and the false neg-

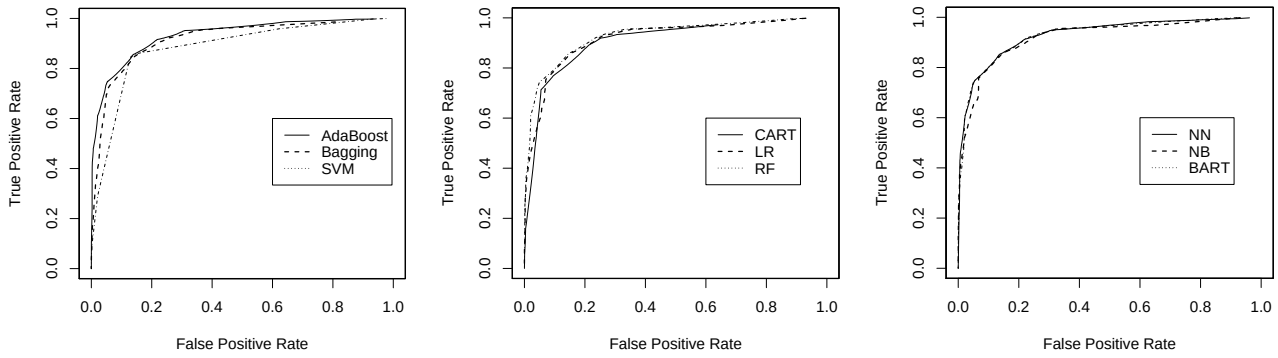


**Figure 2. ROC curves of MLBDMs**

| | FPR(FNR<5.00%) | FNR(FPR<5.00%) |
|---|---|---|
| AdaBoost | 30.65% | 26.15% |
| Bagging | 34.28% | 30.35% |
| SVM | 62.10% | 71.10% |
| CART | 46.24% | 36.77% |
| LR | 33.90% | 41.27% |
| RF | 31.56% | 25.51% |
| NN | 31.56% | 26.48% |
| NB | 31.86% | 35.55% |
| BART | 31.35% | 25.76% |

**Table III. FPR given FNR rate $<$ 5.00%, and FNR given FPR rate $<$ 5.00%**

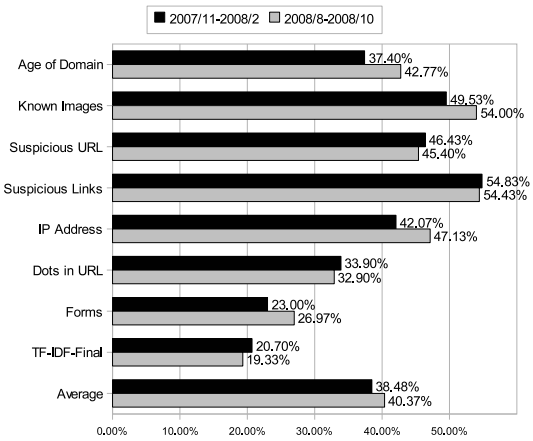


**Figure 4. Error Rate of each heuristic in different time period**

ative rate when the false positive rate was less than 5.00%. The lowest false positive rate was 30.65% in the case of AdaBoost, and the lowest false negative rate was 25.51% in the case of RF. This indicated that if novices could accept 30.65% of false positive errors, 95.00% of phishing sites would be blocked as phishing sites. Similarly, if security experts could accept 25.51% of false negative errors, 95.00% sites of legitimate sites would be browsed normally. It is beyond of scope of this paper, however, we need to decide the optimal threshold for each user by both measuring each user's knowledge for the detection of phishing sites and estimating the acceptable level of false positives and false negatives.

We also check if MLBDMs incorporate new heuristics. Developing sophisticated heuristics are also important to detect new phishing sites. In comparison between Figure 1 and 3, the performance of MLBDMs were tended to decrease in the case of using new dataset. We assumed that was caused by increasing the average error rate of heuristics. Figure 4 showed the error rate of each heuristic where black bars denoted error rates in the case of using old dataset, and gray bars denoted that of using new dataset. We observed that average error rate increased from 38.48% to 40.37%. We need to refine these heuristics and/or to develop new heuristics, then verify whether MLBDMs can incorporate that into the existing heuristics, in our future work.

# 8 Conclusion

In this paper, we evaluated the performance of machine learning-based detection methods (MLBDMs) including AdaBoost, Bagging, Support Vector Machines (SVM), Classification and Regression Trees (CART), Logistic Regression (LR), Random Forests (RF), Neural Networks (NN), Naive Bayes (NB) and Bayesian Additive Regression Trees (BART). Because we assumed that the detection method must be accurate and must have adjustment capability, we used $f_1$ measure,

error rate and AUC as performance metrics in the evaluation.

We employed 8 heuristics presented in [24] and analyzed 3,000 URLs, which were composed of 1,500 legitimate sites and the same number of phishing sites, reported on PhishTank.com from November, 2007 to February, 2008.

The result showed that the highest $f_1$ measure was 0.8581 in AdaBoost, followed by NN (0.8570), SVM (0.8562), BART (0.8555), RF (0.8554), LR (0.8548), NB (0.8547), Bagging (0.8527) and finally CART (0.8384). The lowest error rate was 14.15% in AdaBoost, followed by NN (14.21%), SVM (14.29%), RF and BART (14.45%), LR (14.60%), NB (14.69%), Bagging (14.82%), and finally CART (16.37%). The highest AUC was 0.9342 in AdaBoost, followed by BART (0.9321), NN (0.9310), RF (0.9296), Bagging (0.9231), NB (0.9215), LR (0.9172), CART (0.9062), and finally SVM (0.8956). Additionally, we plotted the ROC curve and found that all MLBDMs could achieve both high true positive rates and low false positive rates. We also compared MLBDMs with the traditional detection method. The result showed that AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods outperformed the traditional detection method.

We found the difference in performance among MLBDMs was not so much, however, almost of all evaluation results showed that MLBDMs achieved better performance than the traditional detection method. Accordingly, we concluded that employing machine learning techniques was appropriate for the detection of phishing sites.

## Acknowledgments

This work was a part of "Research and Development on Traceback Technologies in the Internet" sponsored by the National Institute of Information and Communications Technology(NICT).

## References

[1] Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. "A comparison of machine learning techniques for phishing detection". In *Proceedings of eCrime Researchers Summit (eCryme'07)*, Oct 2007.

[2] Alexa Internet, Inc. "Alexa the Web Information Company". Available at `http://www.alexa.com`.

[3] Anti-Phishing Working Group. "Phishing Activity Trends Report - Q1, 2008". Available at `http://www.apwg.com/reports/apwg_report_Q1_2008.pdf`, Aug. 2008.

[4] Balakrishnan, N. *Handbook of the Logistic Distribution*. Marcel Dekker Inc, 1991.

[5] Basnet, R., Mukkamala, S., Sung, A.H. "Detection of phishing attacks: A machine learning approach". *Studies in Fuzziness and Soft Computing*, 226:373–383, 2008.

[6] Breiman, L. "Bagging predictors". *Machine Learning*, 24(2):123–140, 1996.

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

[8] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. *Classification and Regression Trees*. Wadsworth, 1984.

[9] Chipman, H.A., George, E.I., McCulloch, R.E. "BART: Bayesian Additive Regression Trees". Technical report, Department of Mathmatics and Statistics, Acadia University, Jul. 2005.

[10] Cortes, C., Vapnik, V. "Support-vector networks". *Machine Learning*, 20(3):273–297, 1995.

[11] Fette, I., Sadeh, N.M., Tomasic, A. "Learning to detect phishing emails". In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*, May 2007.

[12] Freund, Y., Schapire, R.E. "Experiments with a new boosting algorithm". In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, Jul. 1996.

[13] Freund, Y., Schapire, R.E. "A decision-theoretic generalization of on-line learning and an application to boosting". *Journal of Computer and System Science*, 55(1):119–139, 1997.

[14] Kumar, A. "Phishing - A new age weapon". Technical report, Open Web Application Secuirtry Project (OWASP), 2005.

[15] Miyamoto, D., Hazeyama, H., Kadobayashi, Y. "A Proposal of the AdaBoost-Based Detection of Phishing Sites". In *Proceedings of the 2nd Joint Workshop on Information security*, Aug. 2007.

[16] OpenDNS. "PhishTank - Join the fight against phishing". Available at `http://www.phishtank.com`.

[17] Pan, Y., Ding, X. "Anomaly based web phishing page detection". In *Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference(ACSAC'06)*, Sep. 2006.

[18] Phelps, T.A., Wilensky, R. "Robust hyperlinks: Cheap, everywhere, now". In *Proceedings of the 8th International Conference on Digital Documents and Electronic Publishing, the 5th International Workshop on the Principles of Digital Document Processing, (DDEP/PODDP'00)*, Sep. 2000.

[19] Robichaux, P., Ganger, D.L. "Gone Phishing: Evaluating Anti-Phishing Tools for Windows". Available at `http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf`.

[20] Tally, G., Thomas, R., Vleck, T.V. "Anti-Phishing: Best Practices for Institutions and Consumers". Technical report, McAfee Research, Mar. 2004.

[21] Van der Merwe, A., Loock, M., Dabrowski, M. "Characteristics and responsibilities involeved in a phishing attack". In *Proceedings of the 4th International Symposium on Information and Communication Technologies (ISICT 2005)*, Jan. 2005.

[22] Yahoo!Inc. "Random Yahoo Link". Available at `http://random.yahoo.com/fast/ryl`.

[23] Zhang, Y., Egelman, S., Cranor, L., Hong, J. "Phinding Phish: Evaluating Anti-Phishing Tools". In *Proceesings of the 14th Annual Network and Distributed System Security Symposium (NDSS'07)*, Feb. 2007.

[24] Zhang, Y., Hong, J., Cranor, L. "CANTINA: A Content-Based Approach to Detect Phishing Web Sites". In *Proceesings of the 16th World Wide Web Conference (WWW'07)*, May 2007.