# Malicious URLs and Attachments Detection on Lexical-based Features using Machine Learning Techniques

by

**Yuanxi Zeng**
B.Eng, University of Electronic Science and Technology of China, 2014

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

**MASTER OF ENGINEERING**

in the Department of Electrical and Computer Engineering

# Supervisory Committee

Malicious URLs and Attachments Detection on Lexical-based Features using Machine Learning Techniques

by

Yuanxi Zeng
B.Eng, University of Electronic Science and Technology of China, 2014

**Supervisory Committee**

Dr. Issa Traoré, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Tao Lu, Department Member
(Department of Electrical and Computer Engineering)

# ABSTRACT

Email is one of the prime sources of cyber-attacks against Internet users. Attackers often use social engineering in order to encourage recipients to click on a link which refers the user to a malicious website or opens a malicious attachment. The recipients may have confidential information stolen and suffer financial loss. In this report, we use two lexical models to detect malicious emails. The models extract solely lexical features from embedded URLs in email content and attachment filenames. The feature extraction does not require external services or tools, thereby it meets the need for real-time detection. The lexical features are used in conjunction with machine learning techniques. Five classifiers are used to test on our dataset of URLs and attachments. The experimental results show that Gradient Boosting Decision Tree outperforms all the other classifiers. It achieves an encouraging accuracy of 90.71% and 94.36% for URL model and attachment model. We can conclude that the lexical models are helpful in malicious email detection. However, we demand an attachment dataset of much more diversity and more features to be added for attachment model.

# Table of Contents

# List of Tables

# List of Figures

# ACKNOWLEDGEMENTS

First, I would like to thank my Supervisor Dr. Issa Traoré, for mentoring, support, encouragement, and patience. Thank you for your providing valuable guidance, insightful advice thorough my graduate study.

My deepest gratitude goes to my parents for supporting me financial and emotional support with your unconditional love. Thank you for understanding me and encouraging me all the time. It is your selfless dedication to my growth that helps me make it this far.

# Chapter 1. Introduction

## 1.1 Background

In contemporary society, the Internet is playing a vital role in almost every aspect of human life and email is one means of information exchange among Internet users. While the email has become a necessary utility in everyday life, it also exposes people to a diversity of cyber-attacks which can have a serious impact on an information society. Malicious email remains one of the most significant and ongoing computer security threats that we face. Attackers use a variety of email-based attacks to deliver malware, lure victims to malicious websites and steal confidential information. They use social engineering in order to encourage recipients to click on a link which refers the user to a malicious website or open a malicious attachment. Malicious email authors constantly develop new, or at least different ways to deceive and attack the victims. Although the malicious payloads found in emails frequently change, the vast majority of cybercriminals use three basic strategies: links to malicious web pages, malicious attachments, enticements to perform transactions.

A malicious email may contain phishing URL links which will direct the user to a phishing webpage. Phishing is an attempt by attackers to steal confidential or sensitive information such as bank account password, credit card information from unsuspecting victims for identity theft, financial gain and other fraudulent activities by disguising a malicious website as a legitimate one [1]. Phishing can also be referred as an automated identity theft, which takes the advantage of the human nature and the Internet to trick millions of people out of large amounts of money. The phishing websites generally have alike page layouts, blocks and fonts to mimic legitimate webpages in an endeavor to bait web users to divulge their personal details such as username and password. It is relatively a current web crime as compared with virus, hacking and remains an ominous threat to client and business around the world.

Malicious email attachments are also a major threat to cyber security. There has been a sharp increase in the use of email attachments as a way to launch an attack. Attackers attach files to email that can install malware capable of destroying data and stealing information. Some of these infections can allow the

attacker to take control of the user's computer, giving attackers access to the screen, capture keystrokes, and access other network systems. According to a Verizon Enterprise report 66% of all malware was installed via malicious email attachments [2]. These attacks have included as Zeus, Cryptolocker, and Dunihi amongst many others.

Since many email systems automatically block obvious malicious programs, such as .exe files, attackers conceal a piece of software called an exploit inside other types of commonly emailed files – Microsoft Word documents, ZIP or RAR files, Adobe PDF documents, or even image and video files. The exploit takes advantage of software vulnerabilities and then downloads the intended malicious software, called a payload, to the computer. Attackers can also embed a malicious macro in the document and use social engineering to trick the user into clicking the "Enable Content" button that will allow the macro to run and infect the victim's computer.

### 1.2 Objective and Contribution

The objective of this report is to give a method of malicious email detection by embedded URLs and attachments. Specifically, we manually select and extract a number of lexical features from URLs and attachment filenames. The substance of page the URL indicates and content of attached files are not required in our method. Since solely lexical features are utilized for classification, the proposed method is fast and convenient in feature extraction. Also, no external services are required for classification. Then, machine learning algorithms are utilized to distinguish malicious (phishing) URLs from legitimate ones, and spam attachments from legitimate ones by the lexical feature values. The methods proposed in this report will contribute to a more secure system combined with other detecting strategies on other properties of emails.

### 1.3 Report Outline

The structure of the rest of the report is as follows:

· Chapter 2 presents the lexical feature model of URL and outlines machine learning algorithms for classification. Experimental results of classification based on lexical features and evaluation metrics are given.

· Chapter 3 presents the lexical feature model of attachment filenames and experimental results.

· Chapter 4 makes concluding remarks and discusses future work.

# Chapter 2. Phishing URL Detection Model

In this chapter, we present the proposed phishing URL detection model by describing the lexical feature model and classification techniques, and by presenting the experimental evaluation approach and results.

## 2.1 URL lexical feature extraction

In this report, we extract lexical features from target URLs for classification. URL is a formatted text string utilized by internet users to identify a resource on the Internet. URL string consists of three elements: network protocol, host name and path. The lexical based feature extraction is used to shape a database of feature values. Lexical features are the textual properties of the URL itself, not the substance of the webpage it indicates. It can be directly extracted from URL string. It includes the length of the host name, length of the URL, the number of dots, presence of suspicious characters such @ symbol, hexadecimal characters and other special binary characters (such as '.', '=', '$', '^') either in the host or path name, etc.

Based on the heuristics, a number of lexical features that can differentiate phishing URLs from benign ones are selected for classification. Reasons why some features are extracted are explained as follows.

- Long URL to hide the suspicious part

Phishers tend to use long URL to hide the doubtful part in the address bar. We examine the length of URL and its hostname.

- Multiple subdomain

Domain names are used to ascertain a unique identity. Assume we have the following link: http://www.hud.ac.uk/students/. A domain name might include the country-code top-level domains (ccTLD), which in our example is "uk". The "ac" part is shorthand for "academic", the combined "ac.uk" is called a second-level domain (SLD) and "hud" is the actual name of the domain. A phishing URL tend to use multiple subdomain to trick users into believe it is a legitimate link. For example, "http://www.paypal.com.uk.webscr.poweringnetworks.com" may seem to be

a legitimate link of PayPal. Thus, the number of dots in hostname is taken into account.

- Using IP address

If an IP address is used as an alternative of domain name in the URL, such as "http://125.98.3.123/fake.html", then it is very suspicious. Because it is unlikely a legitimate webpage to use IP address instead of domain name.

- Using URL Shortening Services "TinyURL"

URL shortening is a method on the World Wide Web in which a URL may be considerably smaller in length and still lead to the required website. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL. For example, the URL "http://portal.hud.-ac.uk/" can be shortened to "bit.ly/19DXSk4". We will examine whether shortening service is utilized in URL.

- URL containing "@" symbol

Using "@" is another way used by phishers to write a URL that appears legitimate but actually lead to different pages. Presence of @ symbol in the URL indicates that, all text before @ is comment. Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol. For example, "http://www.uvic.ca@www.ubc.ca" actually leads to "www.ubc.ca".

- Redirecting using "//"

The existence of "//" within the URL path means that the user will be redirected to another website. The nomenclature "=http://" or "=https://" allows the redirection attack. An example of such URL is "http://www.google.com/-url?q=http://www.badsite.com". This URL would refer a user from one site (in this case, google.com) to another site, badsite.com. Occurrence of multiple ".com" in URL is also suspicious and may lead to the phishing attack by the means of URL redirection. We examine number of "//" and ".com" in URL's path.

- Suspicious words contained

Many phishing URLs are found to contain eye-catching word tokens (e.g., login, signin, confirm, free, etc.). Whether these blacklisted words are contained in URLs

can facilitate detect phishing URLs. The suspicious words we chosen are "confirm", "account", "banking", "secure", "webscr", "login", "signin", "paypal" and "free" which will be verified if they appear in URL.

- Adding Prefix or Suffix Separated by (-) to the Domain

The hyphen symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For an example http://www.Confirme-paypal.com. We examine number of hyphens in hostname.

We summarize the chosen 26 lexical features in Table 2.1. A number of the features are already used for classification in [3][4].

Table 2. 1: Selected Lexical Features

| **Feature Description** | **Feature Type** |
|---|---|
| Length of total URL string | Integer |
| Number of dots in URL | Integer |
| Existence of suspicious words | Boolean |
| Length of hostname | Integer |
| Number of digits in hostname | Integer |
| Number of hyphens in hostname | Integer |
| Number of dots in hostname | Integer |
| Number of dots in subdomain | Integer |
| If hostname is IP | Boolean |
| Longest token length in hostname | Integer |
| Length of path of URL | Integer |
| Presence of “@” in URL | Boolean |
| Presence of double slash in path | Boolean |
| Number of “.com” in path | Integer |
| Number of “-” in path | Integer |
| Number of “/” in path | Integer |
| Number of “=” in path | Integer |
| Number of “;” in path | Integer |
| Number of “?” in path | Integer |
| Number of “&” in path | Integer |

| | |
|---|---|
| Number of "," in path | Integer |
| Presence of parameter part | Boolean |
| Number of queries | Integer |
| Presence of fragment part | Boolean |
| Length of longest token of subdirectory | Integer |
| If URL shortening service is used | Boolean |

Values of Boolean features will be converted to numeric values, with "true" to 1 and "false" to 0. To ensure equality between features, all numeric values were normalized, so their values lie between 0 and 1.

## 2.2 Classification Methods

Machine learning is a widely utilized technology in many areas of modern society, including information security. In this section, we outline five popular supervised machine learning algorithms which will be used for classification between phishing and benign URLs. The theory of five algorithms, which are Logistic Regression[5], Support Vector Machine[6], Decision Tree[7], Random Forest[8], Gradient Boosting Decision Tree[9], are presented. These algorithm implementations are provided by scikit-learn, a free software machine learning library for the Python programming language, and will be used for classification in the next section.

### 2.2.1 Logistic Regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 to 1. Logistic regression is similar to a linear

regression, but the curve is constructed using the natural logarithm of the "odds" of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits. It is defined as:

$$\frac{1}{1 + e^{-x}}$$

Where $e$ is the base of natural logarithms and $x$ is the actual numerical value to be transformed. Input values $x$ are combined linearly using weights or coefficient values to predict an output value $p$. Below is a simple example logistic regression equation:

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$

Where $p$ is the predicted output, $b_0$ is the bias or intercept term and $b_1$ is the coefficient for the single input value $x$. Each column in the input data has an associated $b$ coefficient (a constant real value) that must be learned from training data.

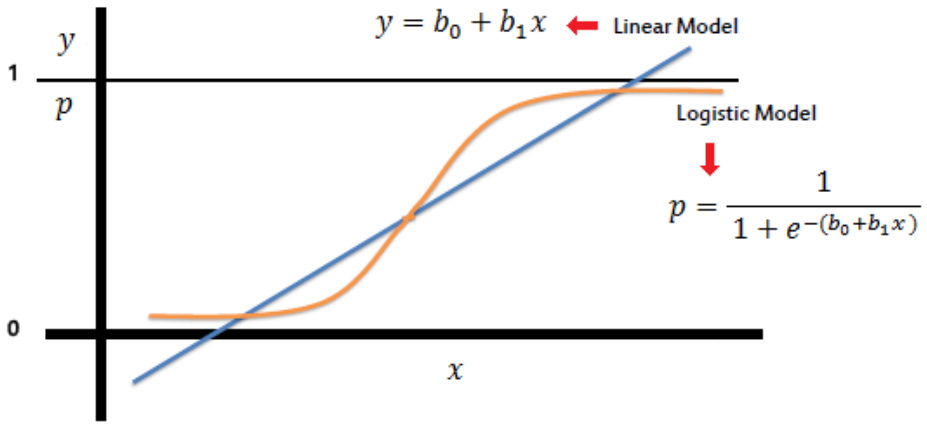



Figure 2. 1: Logistic regression model

In the logistic regression the constant $b_0$ moves the curve left and right and the slope $b_1$ defines the steepness of the curve. Logistic regression can handle any number of numerical and/or categorical variables, so we extend the simple logistic regression equation to multiple predictors:

$$p = P(Y = \text{interested outcome} | X = x, \text{a specific value})$$
$$= \frac{1}{1 + e^{-(b_0 + b_1 x_1 + \cdots + b_k x_k)}}$$

If $p > 0.5$ then classification outcome will be 1 (positive category), otherwise 0 (negative category).

By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$odds = \frac{p}{1-p} = \exp(b_0 + b_1 x_1 + \cdots + b_n x_n)$$

The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.

Logistic regression uses maximum likelihood estimation (MLE) to obtain values of the model coefficients **b** = {$b_0$, $b_1$, …, $b_n$} which maximize the probability of obtaining the data set. The likelihood function is used to estimate the probability of observing the data, given the unknown parameters ($b_0$, $b_1$, …, $b_n$). A "likelihood" is a probability that the observed values of the dependent variable may be predicted from the observed values of the independent variables.

In logistic regression, we observe binary outcome and predictors, and we wish to draw inferences about the probability of an event in the population. Suppose in a population from which we are sampling, each individual has the same probability $p$, that an event occurs. For each individual in our sample of size $n$, $Y_i$ =1 indicates that an event occurs, otherwise, $Y_i$ =0. The observed data are $Y_1, \ldots,$ $Y_n$ and $X_1, \ldots, X_n$. The joint probability of the data (the likelihood) is given by

$$L = \prod_{i=1}^{n} p(y|x)^{Y_i} (1 - p(y|x))^{1-Y_i} = p(y|x)^{\sum_{i=1}^{n} Y_i} (1 - p(y|x))^{n - \sum_{i=1}^{n} Y_i}$$

Natural logarithm of the likelihood is

$$l = \log(L) = \sum_{i=1}^{n} Y_i \log[p(y|x)] + (\text{n} - \sum_{i=1}^{n} Y_i) \log[1 - p(y|x)]$$

In which

$$p(y|x) = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + \cdots + b_k x_k)}}$$

Estimating the coefficients **b** is done using the first derivatives of log-likelihood, and solving them for the coefficients. For this, iterative computing is used. An arbitrary value for the coefficients (usually 0) is first chosen. Then log-likelihood is computed and variation of coefficients values observed. Reiteration

is then performed until maximization of $l$ (equivalent to maximizing $L$). The results are the maximum likelihood estimates of **b**.

After coefficients **b** is obtained, regression equation can be used to predict for test dataset. If $p > 0.5$ then classification outcome will round off to 1 (positive category), otherwise 0 (negative category).

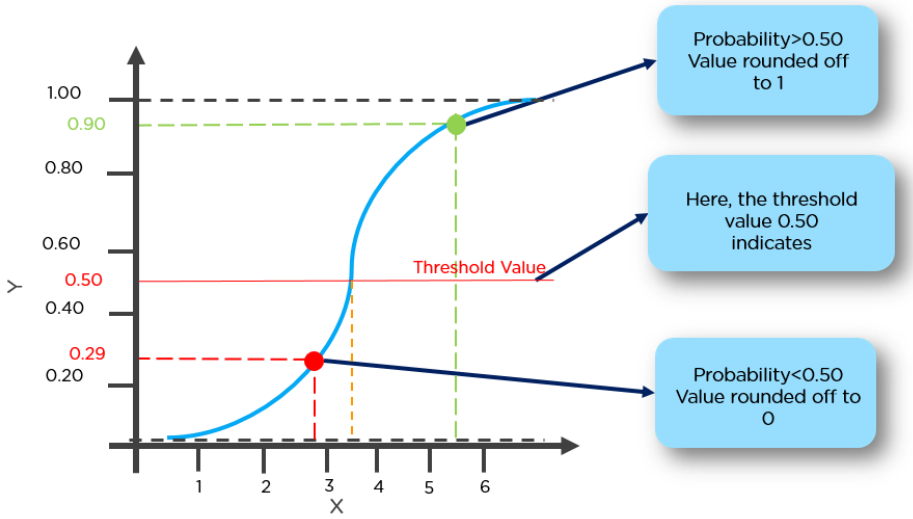



Figure 2. 2: Prediction using logistic regression

In order to reduce overfitting in logistic regression, a penalty is applied to increase the magnitude of parameter values. This strategy is called regularization. Inverse of regularization strength will be used as tuning parameter in experiment.

### 2.2.2 Support Vector Machine

A Support Vector Machine (SVM) is a classifier that makes predictions based on decision boundaries drawn by hyperplanes. A hyperplane is a frontier that separates data from different classes. In a two-dimensional feature space, the hyperplane is simply a line that splits a plane into two distinct parts for each class respectively. In a three-dimensional features space, it will be a plane that splits the space. SVM first creates a $n$-dimensional space ($n$ is the size of feature set) and maps each data sample to a point in the space, whose coordinators are the value of corresponding features. Then it tries to find the optimal hyperplanes that segregate the points by their class. The distance between the hyper-plane and the closest data points is referred to as the margin. In SVM, optimal hyperplanes are defined as the planes which differentiate the classes with the largest margin

possible. In simple terms, they maximize the distances between the nearest data points in all classes. Thus, only the closest points to the hyper-plane are relevant in defining the hyper-plane and in the construction of the classifier. These points are called the support vectors.

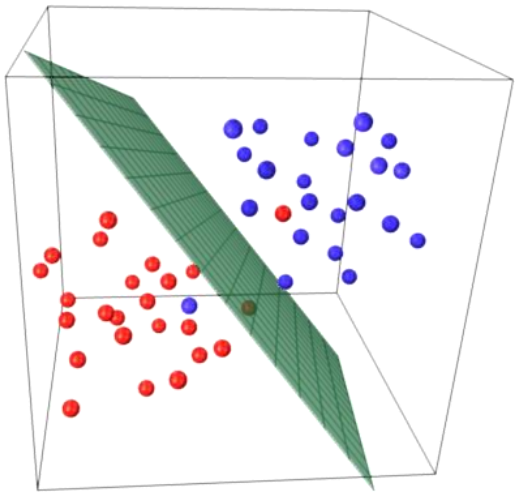


Figure 2. 3: A hyperplane separates data samples in 3D space

In practice, real data is messy and cannot be separated perfectly with a hyperplane. The constraint of maximizing the margin must be relaxed. This change allows some points in the training data to violate the separating hyperplane. An additional set of coefficients are introduced that give the margin wiggle room in each dimension.

A penalty parameter is introduced as a tuning parameter called $C$ that defines the magnitude of the wiggle allowed across all dimensions. The $C$ parameter defines the amount of violation of the margin allowed. The smaller the value of $C$, the lesser the sensitivity of an algorithm to the training data and it tries to find separating plane with largest margin which results in lower variance but higher bias. On the other hand, the larger the $C$ is, the higher the sensitivity of an algorithm is to the training data because it tries to separate all classes correctly which results in higher variance and lower bias. Another important parameter of SVM is γ. Intuitively, the γ parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The γ parameter can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

In most real cases, data is not linearly separable. Thus, the SVM algorithm is implemented in practice with a kernel. The idea stems from the fact that if the data cannot be partitioned by a linear boundary in its current dimension, then

projecting the data into a higher dimensional space may make it linearly separable. This process is illustrated in Figure 2.4 and Figure 2.5.

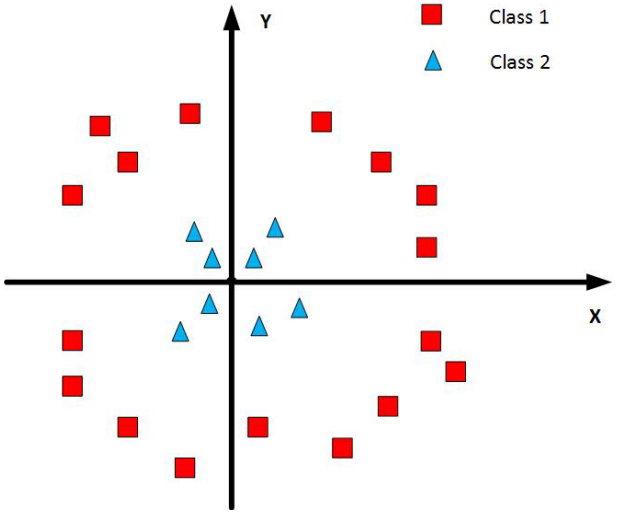



Figure 2. 4: Class 1 and 2 are not linearly separable

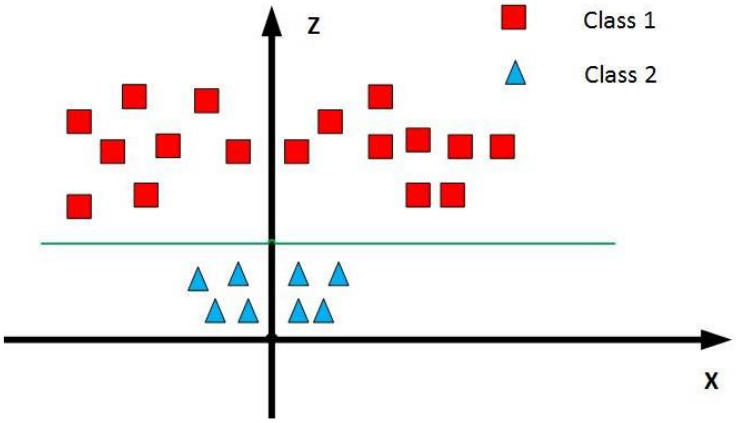



Figure 2. 5: Class 1 and 2 are linearly separable after projecting to higher dimension

Type of kernels:

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma > 0$:
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r), \gamma > 0.$

  where $\mathbf{x}_i$ and $\mathbf{x}_j$ are observed data points.

Here, controlling parameter is $\gamma$ for all the kernels, except linear. RBF is the most popular kernel choice in SVM.

### 2.2.3 Decision Tree Classifier

Decision tree classifier is built on a decision tree model. A decision tree is a graphical representation of all the possible results to a decision based on certain conditions. It is called a decision tree because it starts with a root node, then it branches off into a number of solutions, which gives it a tree-like shape. A decision tree is composed of three basic elements: decision nodes, branches and leaf nodes. Every decision tree can only have a single one root node, which is the first decision node, acting as the starting point. The decision nodes specify an attribute of data samples and contain certain rules to be met, while leaf nodes contain decision or classification information. A branch corresponds to one of the attribute outcomes of a decision node. By connecting nodes with directed paths, the sequence of branches from root to a leaf shows the process of a specific decision being made. Each decision nodes have a least 2 branches stretching out of them. The number of branches from a node is equals to the total number of possible outcomes for the rule in the node. Figure 2.6 shows an example decision treed created from sample data.

In decision trees, two major phases should be ensured:

1. *Building the tree*. The process of constructing a decision tree is basically a divide-and-conquer process. Based on given training data, the decision tree classifier recursively splits the data set into multiple subsets in accordance with the most significant differentiator over all the features of training data, and terminates when either all subsets are homogeneous or certain criteria are fulfilled.
2. *Classification*. In order to classify a new instance, we start by the root of the decision tree, then the feature specified by the node is tested. The result of the test allows to move down the tree branch relative to feature value of the given instance. This process is repeated until a leaf is encountered. This instance is then classified in the same class characterizing the leaf node.


Figure 2. 6: Example of a Decision Tree built on sample data

The problem here is how to choose the best feature to make splits for each decision node during construction of the decision tree. Several algorithms have been developed in order to find the split that generates the most homogeneous subsets, such as ID3, C4.5 and CART.

ID3 algorithm creates a multiway tree, finding for each node the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalize to unseen data.

C4.5 is the successor of ID3 and removes the restriction that features must be categorical by dynamically defining discrete attributes (based on numerical variables) that partitions the continuous feature value into a discrete set of intervals. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

The major drawback of decision tree is its tendency to overfitting. Tuning parameters and pruning are introduced to reduce decision tree complexity. For instance, minimum samples for a node split, minimum samples for a leaf node, maximum depth of tree, maximum number of terminal nodes, maximum features to consider for a split are some popular tuning parameters for a decision tree classifier. In addition, pruning is a better way to improve the performance of a decision tree. Pruning suggests to remove the branches which adopt less

important features. Reduced error pruning and cost complexity pruning are two widely used pruning options.

### 2.2.4 Random Forest Classifier

Random forest (RF) classifier is an ensembled algorithm based on the decision tree model. The goal of ensemble methods is to combine a collection of base prediction model in order to improve classification accuracy over a single estimator. Ensembling techniques are further classified into Bagging and Boosting. Random Forest is a bagging algorithm that use decision tree as base predicator. Random forest classifier consists of multiple decision trees, which are introduced in the previous section. After a large number of trees is generated, they vote for the most popular class together. In general, the bagging algorithm considers the function estimation *F* in form as follows:

$$F(\mathbf{x}) = \text{avg}\big(f_i(\mathbf{x})\big), \qquad i = 0,1, \dots, m$$

where $f_i(\mathbf{x})$ is base predictor (decision tree in RF), $m$ is the number of predictors.

Random forest adds additional randomness to the model while growing the tree. Only a random subset of the entire training samples is adopted as the training set for a single tree, which is called bootstrapping, so that all the trees are little different from each other. Also, only a random subset of features is used in tree growing as well in order to make each tree less correlated to each other and reduce variance. This results in a wide diversity that generally results in a better model. Every tree should use same size of training samples and features space in decision tree building. The method of tree growing is the same as described in the previous section. Additionally, trees in random forest can have different weights. By assigning more accurate trees more weights and less weights to accurate ones, the performance of the algorithm is further improved.

Random forest classifier is based on tree model. Thus, it naturally has all the tuning parameters associated with decision trees. Another important tuning parameter for random forest is the number of trees in forest.

The underlying principle of ensembled algorithms is that although one single model may be limited under certain circumstances, a group of models tends to be much powerful. While a single decision tree is inclined to over-fitting and high variance, a random forest substantially increases the model stability.

### 2.2.5 Gradient Boosting Decision Tree

Gradient Boosting Decision Tree (GBDT) is also an ensemble algorithm based on decision tree model like Random Forest. However, it is a boosting algorithm as its name indicates. Unlike a bagging algorithm in which predicators are independent, boosting algorithm makes predicators sequentially. Gradient boosting considers the function estimation in an additive form as follows:

$$F_m(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x})$$

where $F_i(\mathbf{x})$ is the prediction function after the $i$th iteration in boosting process, and $f_i(\mathbf{x})$ is the decision tree generated in $i$th iteration. $F_m(\mathbf{x})$ will be the final prediction function of GBDT, where m is the number of base predictors (decision trees).

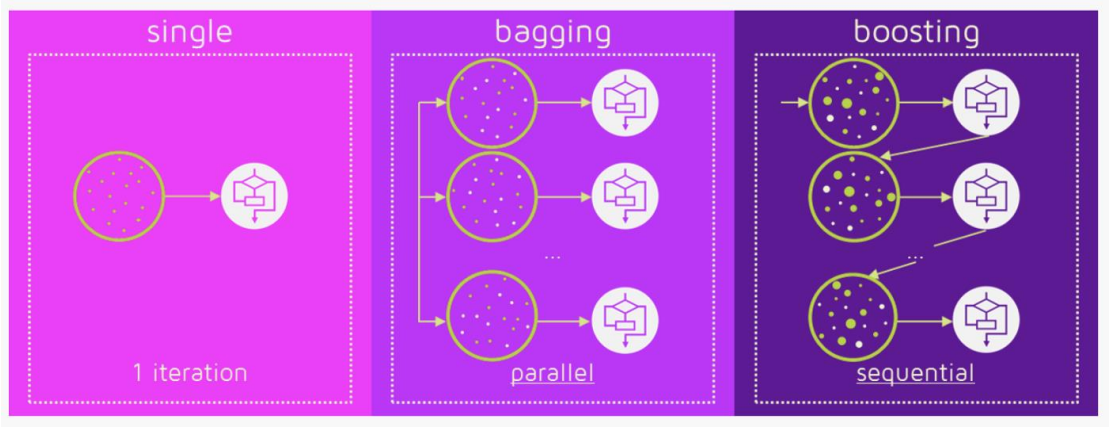



Figure 2. 7: Comparison of single, bagging and boosting classifiers

The objective of any supervised learning algorithm is to define a loss function and minimize it. By using gradient descent and updating our predictions based on a learning rate, we can find the values where the loss is minimum. Thus, we are basically updating the predictions such that the sum of our residuals is close to zero or minimum and predicted values are sufficiently close to actual values. The intuition behind gradient boosting algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better.

The steps of GBDT building is briefly described as follows:

1. Build a weak decision tree $f_1(x)$ on training data.
2. Calculate error residuals $r$, which is actual target value minus predicted target value.

3. Build a new decision tree model $f_2(x)$ on error residuals as target variable with same input variables. Then the prediction function will be $F_2(x)= f_1(x)+ f_2(x)$ for the next iteration.
4. Repeat step 1~3 until $m$ decision trees are built, $F_m(x)= f_1(x)+ f_2(x)+...+ f_m(x)$ is taken as the prediction function of GBDT.

## 2.3 Evaluation and Result Analysis

In this section, we will experiment with different classifiers introduced in 2.2 on URL lexical feature model presented in 2.1. A URL dataset containing both phishing and legitimate URLs are used for training and testing. Furthermore, different metrics for performance evaluation are described. Then, we will analyze the classification results. The classifiers and metrics computation are implemented in Python, together with the powerful machine learning tool scikit-learn[10].

### 2.3.1 Dataset

The URL dataset is retrieved from two online sources, one for benign URLs and the other for malicious URLs. We collected 15000 benign URLs and 15000 phishing URLs, that is 30000 samples in total. The phishing URLs are taken as positive samples and benign URLs are taken as negative samples.

The benign URL dataset was retrieved from the DMOZ Open Directory[11] website. The DMOZ Open Directory is a free, open source and community-based website that allows users to browse the directory and suggest URLs. In addition, other users can volunteer to edit the directory, which allows for some validation of URLs prior to them being added, modified or deleted in the DMOZ.

The malicious URL dataset was retrieved from the PhishTank[12] website and all URLs were verified as malicious at the time of retrieval. PhishTank is a free community website that allows users to submit, verify, track and share phishing URL data.

### 2.3.2 Cross-Validation

Cross-validation (CV) is a resampling procedure used to evaluate the performance of machine learning models on a limited data sample. It is commonly

used in applied machine learning to select and compare a model to a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

In this project, we will use a basic approach in CV called *k*-fold. *k* refers to the number of groups that a given data sample is to be split into. The following procedure is followed for each of the k "folds":

- A model is trained using samples from *k*-1 folds as training data
- The resulting model is validated on the remaining fold of the data

The performance of the model is measured by the average of the *k* test scores generated in the iteration. Although *k*-fold CV is computationally expensive, it has the advantage of making use of most of the data while preserving its original features, thus it is also widely used in parameter tuning and performance evaluation.

### 2.3.3 Evaluation Metrics

For a binary classification problem like the one we are dealing with, consider the matrix

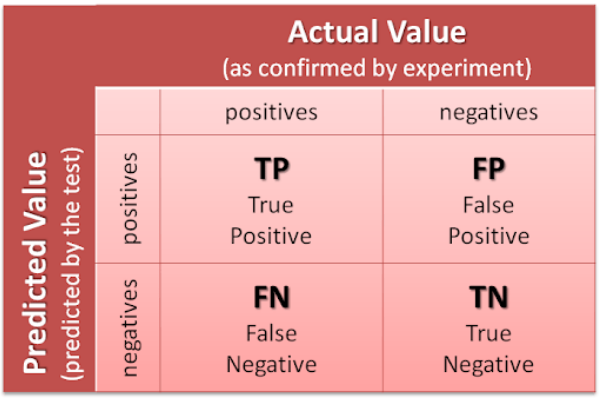



Figure 2. 8: Confusion Matrix

This is called confusion matrix and shows all four possible outcomes of a binary classification. The detail of the evaluation metrics are as follows:

**True Positives** (*TP*): The number of phishing URLs identified as phishing

**True Negatives** (*TN*): The number of benign URLs identified as benign

**False Positives** (*FP*): The number of benign URLs misidentified as phishing

**False Negatives** (*FN*): The number of phishing URLs misidentified as benign

Given the above confusion matrix, a number of metrics are commonly used to evaluate classification performance, which are defined below:

**Accuracy** (*ACC*) is the rate of correctly identified samples among all samples, and defined as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

**Precision** ($p$), or positive predictive value, measures the rate of true positive samples among all identified positive samples, and is defined as:

$$p = \frac{TP}{TP + FP}$$

It is important to note that misclassifying a phishing URL may have a different impact than misclassifying a legitimate URL, so we report separately the rate of true positives and true negatives.

**True Positive Rate** (*TPR*), also known as sensitivity or recall, measures the rate of correctly-identified positive samples among all true positive samples, and is defined as:

$$TPR = \frac{TP}{TP + FN}$$

**False Positive Rate** (*FPR*), also called fall-out, measures the rate of false samples among all identified positive samples, and is defined as:

$$FPR = \frac{FP}{TN + FP}$$

**True Negative Rate** (*TNR*), also called specificity, measures the rate of correctly-identified negative samples among all true negative samples, and is defined as:

$$TNR = \frac{TN}{TN + FP} = 1 - FPR$$

**F1 score** (F1) is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account. F1 is usually more useful than accuracy, especially the class distribution is uneven. F1 score is defined as:

$$F1 = \frac{2 \cdot p \cdot TPR}{p + TPR} = \frac{2TP}{2TP + FP + FN}$$

Another metric to evaluate the classifiers is standard deviation(std) of k-fold cross-validation. It is the standard deviation of the k test scores from k-fold cross-validation. A low standard deviation indicates a more stable classification model to training data.

The metrics mentioned above will be used to evaluate performance of phishing URL classification in this project.

### 2.3.4 Experimental Results

Using the proposed lexical feature model, we encode our URL samples into feature vectors. Then the feature vectors are passed to the five machine learning algorithms, which are Logistic Regression, SVM, Decision Tree, Random Forest and GBDT, for phishing URLs classification. 10-folds cross validation is used to evaluate the classification performance. The dataset is divided into ten different parts, and nine parts out of ten will be utilized to train the classifier. Then the trained classifier will test the tenth part. This process is completed ten times and at the end of training and testing phase, all of the parts are used as both training and testing data.

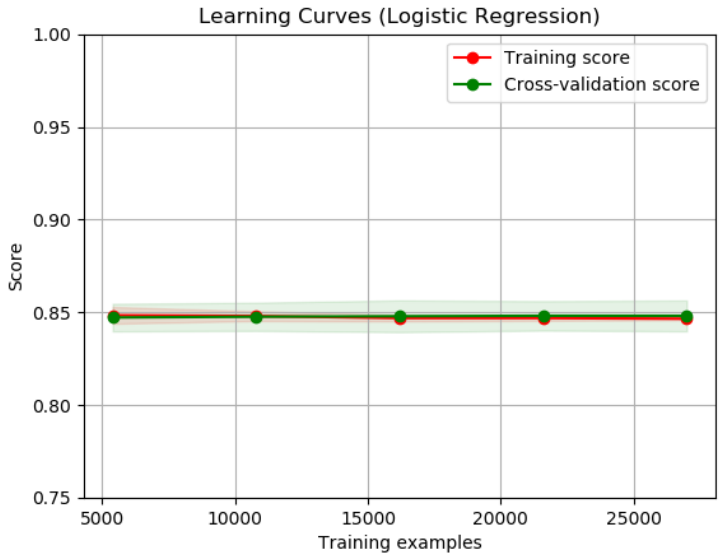
**Logistic Regression**

For logistic Regression, we choose "*penalty*" = "*l1*", inverse regularization strength "*C*" =10 as tuning parameters. The experimental results of 10-fold CV for logistic regression are shown in Table 2.2.

Table 2. 2 Performance results for Logistic Regression

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **Logistic Regression** | 84.64% | 87.41% | 80.95% | 88.34% | 84.05% |

Figure 2. 9 Learning Curves of Logistic Regression



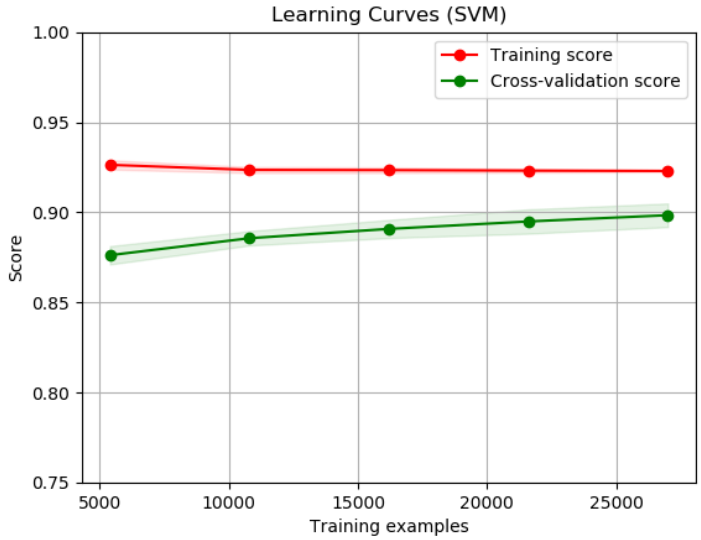

**Support Vector Machine**

For SVM, we choose "*kernel*" = "*rbf*", penalty coefficient "C" =1, "*gamma*" = 1 as tuning parameters. The experimental results of 10-fold CV are presented in Table 2.3.

Table 2. 3 Performance results for SVM

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **SVM** | 89.80% | 90.80% | 88.56% | 91.03% | 89.67% |

Figure 2. 10 Learning Curves of SVM



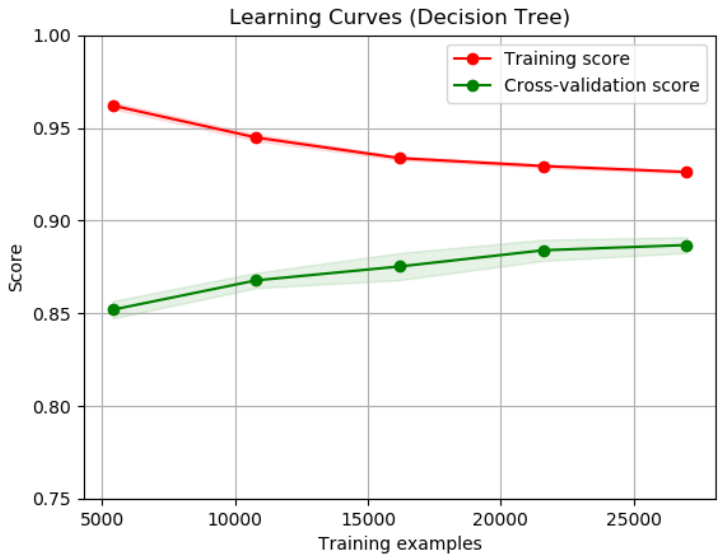

**Decision Tree**

For Decision Tree, we choose the minimum number of samples required to split an internal node "*min_samples_split*"=0.001 as the tuning parameter . The experimental results of 10-fold CV are presented in Table 2.4.

Table 2. 4 Performance results for Decision Tree

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **Decision Tree** | 88.77% | 89.72% | 87.57% | 89.97% | 88.63% |

Figure 2. 11 Learning Curves of Decision Tree



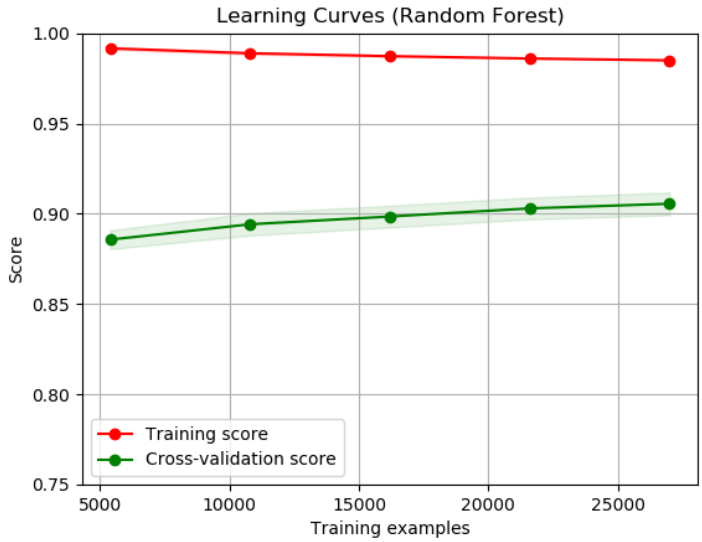

**Random Forest**

For Decision Tree, we choose the number of base estimator "*n_estimators*"=70 as the tuning parameter. The experimental results of 10-fold CV are presented in Table 2.5.

Table 2. 5 Performance results for Random Forest

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **Random Forest** | 90.52 % | 91.38 % | 89.48 % | 91.56 % | 90.42% |

Figure 2. 12 Learning Curves of Random Forest



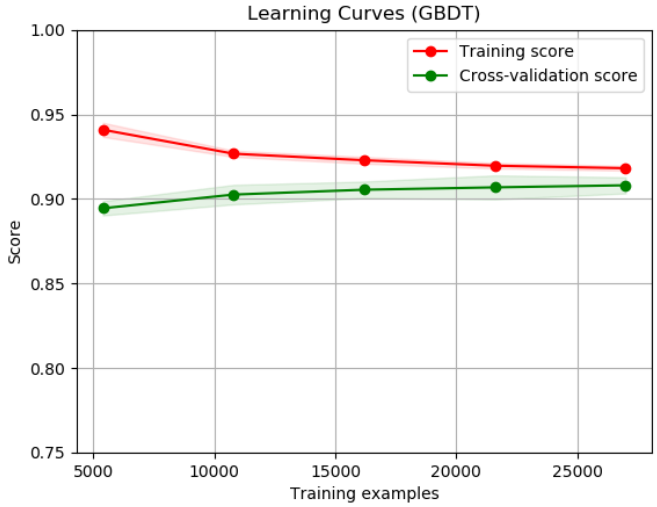
**Gradient Boosting Decision Tree**

For GBDT, we choose "*learning_rate*"=0.5 and "*n_estimators*"=120 as the tuning parameters. The experimental results of 10-fold CV are presented in Table 2.6.

Table 2. 6 Performance results for GBDT

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **GBDT** | 90.71% | 91.92% | 89.27% | 92.16% | 90.58% |

Figure 2. 13 Learning Curves of GBDT




Summarizing the previous results, Table 2.7 provides a straightforward comparison between these five classifiers in terms of accuracy, precision, TPR, TNR. Table 2.8 presents the standard deviation of 10-fold cross-validation of the 5 classifiers.

Table 2. 7 Performance comparison between the different classifiers

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **Logistic Regression** | 84.64% | 87.41% | 80.95% | 88.34% | 84.05% |
| **SVM** | 89.80% | 90.80% | 88.56% | 91.03% | 89.67% |
| **Decision Tree** | 88.77% | 89.72% | 87.57% | 89.97% | 88.63% |
| **Random Forest** | 90.43% | 91.16% | 89.55% | 91.32% | 90.42% |
| **GBDT** | 90.71% | 91.92% | 89.27% | 92.16% | 90.58% |

Table 2. 8 Standard deviation of 10-fold cross-validation of URL classifiers

| Logistic Regression | SVM | Decision Tree | Random Forest | GBDT |
|---|---|---|---|---|
| $5.91\times10^{-3}$ | $3.65\times10^{-3}$ | $6.36\times10^{-3}$ | $4.98\times10^{-3}$ | $4.73\times10^{-3}$ |

In comparison, we found that Gradient Boosting Decision Tree outperforms the other classifiers in accuracy (90.71%), precision (91.93%), TPR (89.24%), TNR (92.17%) and F1(90.58%). In GBDT classification, 13391 out of 15000 phishing URLs are correctly identified, and 13824 out of 15000 benign URLs are correctly identified. It also has the second lowest standard deviation. The following classifiers in terms of performance are Random Forest, SVM, Decision Tree and Logistic Regression. Random Forest, SVM, Decision Tree are almost as good as GBDT. Performance of Logistic Regression is poorer than the others, but still it can achieve at least an accuracy of 84.25%.

# Chapter 3. Spam Attachment Detection Model

The structure of this chapter is similar to chapter 2. In this chapter, we present the spam attachment detection model by describing the lexical feature model on attachment filename, and analyze the experimental results. Most spam do not have criminal intent, but attachments in spam messages are in general malicious. So, spam attachment in general falls under the category under phishing where the sender is trying to convince the recipient to perform some actions (i.e. clicking on the attachment) that ultimately could be harmful. Since some techniques and evaluation methods are already introduced in Chapter 2, we will not repeat those in this chapter. Instead, the chapter will introduce only the feature model, and present the evaluation datasets and results.

## 3.1 Attachment Filename Lexical Feature Extraction

Cyber attackers may use email attachments as a way to launch attacks. Even though the attackers can set the filename of attachments at random, attachment names from a same attacker can follow a certain pattern. Thus, lexical features can be helpful in detecting attachments from malicious emails.

We will extract solely lexical feature from the email attachment filenames without looking into content of the files. Table 3.1 shows the 15 lexical features which we extract.

Table 3. 1 Selected 15 lexical features of attachment filenames

| **Lexical Features** |
|---|
| Length of filename |
| Number of English letters |
| Number of digits |
| Number of capital letters |
| Number of spaces |
| Number of dashes/hyphens |
| Number of other characters |
| Number of vowels |
| Ratio of English letters |

| |
|---|
| Ratio of digits |
| Ratio of vowels |
| If the first character is letter |
| If the first character is digit |
| If the last character is letter |
| If the last character is digit |

Similar to the data processing approach used for URL feature model, all feature values will be standardized to numerical values between 0 and 1.

## 3.2 Evaluation and Result Analysis

The machine learning algorithms we used for attachment classification will be the same as those outlined in section 2.2. We will also use the same metrics for performance comparison and evaluation.

### 3.2.1 Dataset

The email attachments are collected from two online resources: Enron Email[13] Dataset and Untroubled Software[14]. 515 legitimate email attachments are retrieved from Enron Email Dataset, which contains more than 500,000 emails from 150 employees of the Enron Corporation. Untroubled Software provides 213 attachments from spam emails. Even though these emails might not do any damage to receiver's computer, it can still be used for classification as malicious attachments do not have any special property in their filename. If spam attachments can be distinguished from legitimate ones, then it is likely for malicious attachments. However, there are drawbacks of the spam attachment dataset. Size of the dataset is not big. On top of that, most of the attachment types are .7z, and they have very similar pattern in name.

### 3.2.2 Experimental Results

Features vectors of data samples will be used to train and test the five classifiers: Logistic Regression, SVM, Decision Tree, Random Forest and GBDT. The results of 10-fold cross-validation using these classifiers are presented in Table 3.2.

For SVM classifier, tuning parameters are "*gamma*"=1, C=10. For Random Forest, tuning parameter is "*n_estimators*"=70. For GBDT, the tuning parameters are "*n_estimators*"=70. Other tuning parameters of classifiers are left as default.

Table 3. 2 Experimental results for all classifiers after 10-fold cross-validation

| | *ACC* | *p* | *TPR* | *TNR* | *F1* |
|---|---|---|---|---|---|
| **Logistic Regression** | 87.50% | 83.15% | 71.83% | 93.98% | 77.08% |
| **SVM** | 92.03% | 94.79% | 76.99% | 98.25% | 84.97% |
| **Decision Tree** | 92.99% | 88.94% | 86.85% | 95.53% | 87.88% |
| **Random Forest** | 94.23% | 93.40% | 86.38% | 97.47% | 89.75% |
| **GBDT** | 94.36% | 92.57% | 87.79% | 97.08% | 90.12% |

Table 3. 3 Standard deviation of 10-fold cross-validation of attachment classifiers

| **Logistic Regression** | **SVM** | **Decision Tree** | **Random Forest** | **GBDT** |
|---|---|---|---|---|
| $8.51\times10^{-2}$ | $6.92\times10^{-2}$ | $3.96\times10^{-2}$ | $4.16\times10^{-2}$ | $4.38\times10^{-2}$ |

As Table 3.2 shows, Random Forest and GBDT outperform the others. Again, GBDT achieves the highest accuracy among all. In GBDT, 187 out of 213 spam attachments are correctly detected, and 15 out of 515 legitimate attachments are falsely identified as spam. It also has the second lowest standard deviation.

# Chapter 4. Conclusion and Future Work

## 4.1 Conclusion

In the current report, we proposed solely lexical feature models of URL and attachment filename for malicious email detection. Since we only extract lexical features from email itself, based on quick static analysis, and their extraction does not require an Internet connection or the use of external services or other tools, the model is suitable for a real-time detection system. We extracted 34 lexical features in URL model and 15 features in attachment model. The feature model was evaluated with five machine learning classifiers on a fixed data set. The classifiers included Logistic Regression, Support Vector Machine, Decision Tree, Random Forest and Gradient Boosting Decision Tree. A total of 30000 URLs were collected for the phishing data set. The phishing data set consisted of 15000 phishing samples and 15000 legitimate URL samples. The attachment data set consisted of 213 spam attachments and 515 benign attachments. 10-fold cross-validation was executed and evaluation metrics were calculated. In order to evaluate the classification performance, the main evaluation metrics were accuracy, precision, true positive rate and true negative rate.

The performance evaluation showed that Gradient Boosting Decision Tree outperformed the others with the highest accuracy and f1 score for URL model and attachment model. The performance is encouraging and shows that the solely lexical feature model can be very helpful in malicious email detection. However, there is still a considerable error rate (around 10%) in the URL feature model. Since the information we extracted from URLs and attachments is merely from the string and URLs and attachment names, and the models do not include more complicated lexical features, the detection performance is limited by this fact. It is worth to put efforts into further research into the model to improve the feature set and combine it with other email features, including body-based features, subject-based features, script-based features, sender-based features, for malicious email detection.

## 4.2 Future Work

As mentioned in chapter 4, the dataset for email attachment requires more samples and diversity. We failed to find a sufficient online dataset which contains a large number of spam attachments. The attachment feature model is still

defective. The performance of classification may be quite different if the dataset is enriched. Apart from enriching the size of dataset, more features can be considered and added to the model to incorporate file format and type. Also, the size of legitimate attachment dataset should be enlarged. The samples in it should be more carefully selected for a good generalization.

On top of that, we can investigate more other machine learning techniques, like Neural Network, AdaBoost. We can even try to combine different machine learning classifiers together to see if it can have a better classification performance. This model does not work along for malicious email filtering, it will be integrated with other properties of emails for a more sophisticated detection. Last but not least, we can try to introduce more complicated lexical features into the model and conduct more robust error analysis to improve the accuracy.

# References


[1]. Gunter Ollmann, The Phishing Guide Understanding & Preventing Phishing Attacks, IBM Internet Security Systems, 2007.

[2]. Verizon Enterprise, https://www.verizonenterprise.com/resources/reports/-2017_dbir_en_xg.pdf.

[3]. Martyn Weedon, Dimitris Tsaptsinos, James Denholm-Price. "Random forest explorations for URL classification", 2017 International Conference On Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA), 2017.

[4]. Anh Le, Athina Markopoulou, Michalis Faloutsos. "PhishDef: URL Names Say It All", IEEE INFOCOM 2011, 2011.

[5]. Hosmer, David W. "Applied Logistic Regression, 3rd Edition" published by New York: Wiley Publishing, 2000.

[6]. Colin Campbell, Yiming Ying. "Learning with Support Vector Machines" published by Morgan & Claypool Publishers, 2011.

[7]. Vili Podgorelec, Peter Kokol, Bruno Stiglic, Ivan Rozman. "Decision Trees: An Overview and Their Use in Medicine", in Journal of Medical Systems, Volume 26 Issue 5, pp. 445-463, October 2002.

[8]. Leo Breiman. "Random Forests", in Machine Learning, Volume 45, Issue 1, pp. 5–32, October 2001.

[9]. Jerome H.Friedman. "Stochastic Gradient Boosting", Computational Statistics & Data Analysis, Volume 38, Issue 4, pp. 367-378, 28 February 2002.

[10]. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine learning in Python", Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[11]. Open Directory Project. http://www.dmoz.org.

[12]. PhishTank. http://www.phishtank.com.

[13]. Bryan Klimt, Yiming Yang. The Enron Corpus: A New Dataset for Email Classification Research. 2004.

[14]. Untroubled Software. http://untroubled.org/.