CS-559: Final Project

Justin Ho

15 December 2020

I pledge my honor that I have abided by the Stevens Honor System. *Justin Ho*

1 The Problem

In the field of computer security, a large problem in the end user space is the problem of social engineering, in which malicious actors try to gain access to some protected resource that requires authentication through various social tactics, like deception. One especially popular avenue of social engineering that has existed throughout the development of the Internet is phishing, in which attackers try to recreate a seemingly legitimate interface for some established platform to potentially deceive end users to enter sensitive credentials to gain access to the user's resources on said platform. A concrete example is in the context of social media phishing, in which attackers attempt to trick users into entering their account credentials into a malicious website the attacker controls that looks like, say, Twitter. If the user falls for the deception, the attacker is successful and the user's account is compromised. Of course, such attacks are not limited in scope to social media, extending across various mediums, like emails to employees of large companies, like ISPs.

While it is not too difficult to the tech-savvy user to pick apart phishing attempts, there has not been much in the way of formally trying to train a machine learning algorithm to classify potential phishing links; this is due in part to not having reliable training sets for this problem. Through the efforts of Rami M. Mohammad, Fadi Thabtah, and Lee McCluskey, however, such a dataset now exists; it is through this training set that I implemented bagged decision trees to automate classification of phishing links.

2 The Data

At a high level, the dataset used to classify phishing links separates the attributes into four distinct categories, all contributing to a total attribute count of 30, with the final 31st attribute being the class attribute, which specifies if the entry is a phishing or legitimate link. The four categories are as follows:

1. Address Bar-based Features: These features are based on several aspects of what appears in the address bar with selected heuristics based on general practices employed by phishing attacks.

2. Abnormality-based Features: These features are based on general abnormalities in the practices employed within in the site, like the use of the `mailto` protocol when a user submits a form.

3. HTML and Javascript-based Features: These features are based on the behavior of the HTML and Javascript have on the site, from the DOM to user actions like clicking.

4. Domain-based Features: These features are based on the domain of the links themselves and its properties, like DNS records.

Every feature within the dataset will have a value of either `1` (legitimate), `-1` (phishing), or `0` (suspicious). Only certain features have `0` available as a value, however. Mohammad, et al., goes into deeper detail on each of the 30 attributes of their dataset in the attached `features.docx` document as well as the attached `.arff` files.

As for how the data is used in my implementation, I use two datasets provided by Mohammad, et al. `data.csv` is the "main" dataset used, in that I use the training data extracted from it to train the trees. I use `test.csv` as an additional testing block; this dataset was labeled as "old" by Mohammad, et al., but I decided to use it as a separate extra test to see how the trees perform. As for the training and test splits, I cut `data.csv` with a 70:30 split for training and testing upon randomization of the matrix, respectively, with `test.csv` being completely reserved for testing.

As for the code itself, it generates a mean and standard deviation for both test sets separately; it additionally does this across several iterations, which each iteration changing the number of trees generated and the number of execution iterations over that number of trees. Specifically, the code increases the number of trees generated (goes from 1 to 500 in large, discrete steps) once that number of trees finishes going through 1 iteration, 5 iterations, and 10 iterations.

Lastly, `data.csv` and `test.csv` were derived by the attached `train.arff` and `test.arff`, respectively, which were the original formats of the datasets.

3 The Results

From the results attached in `results.txt`, we see that accuracy of the test data from `data.csv` is not too interesting, with essentially all the generated trees passing with flying colors; with having 30 features to base predictions on, the trees generally have a good tolerance against false predictions. Of course, the more trees in the ensemble, the more accurate the predictions became, with the beginning iterations having sub-100% accuracy compared to the 100% accuracy of the larger ensembles. However, the interesting results come from the extra testing provided by `test.csv`; the means of the early iterations are spotty compared to the testing done over `data.csv`. The reason why this happens is because `test.csv` has a slightly different feature classification scheme from `data.csv`. Because the early iterations have much smaller ensembles than the later iterations, errors are much more common since these ensembles essentially do not have enough "perspectives" within the ensemble to ensure as many correct predictions consistently, which we see in their rather large standard deviations, with a standard deviation even going as high as 0.25 in the case

of the 10-tree, 5-iterations round. In contrast to the these spotty early iterations, once the rounds reach the 50-tree mark, we see extremely consistent and high-accuracy predictions; of course, these ensembles have many members and thus have a large number of "perspectives" to combine into one prediction, which turns out to be resistant against the different feature classification scheme of `test.csv`.

From these results, it is clear that the number of features in the dataset and the number of trees within an ensemble matter in the quality of bagged decision tree predictions. However, it must be said that the former may not always be the case, as this dataset happened to have all its features be important determining factors in a good prediction, whereas some datasets may have insignificant features; thus, it is the number of relevant features in the dataset that matters. Of course, the number of trees used in this form of learning is of vital importance, as more trees naturally gives better accuracy.