# Classifying benign and phishing URLs

Charles Paulet
ID: 18103197

Pierre Blank
ID: 18105734

Alexandre Lambridinis
ID: 18103138

***Abstract*—This is the report of the first ca645 assignment. Our task is to research and implement a phishing URL classifier and to write a paper detailing your approach and findings. This is the paper.**

## I. Introduction

In this document, we will describe the goals of the project, how we achieved them, what difficulties we encountered and some suggestions for improvement.

## II. The goals of the project

As previously written, the main goal is to research and implement a phishing URL classifier. The research step is of course necessary for the implementation one. We had some vague notions about machine learning but not as much as required by this project. In fact, we had only theoretical knowledge of the field, it was time for us to have a bit of practice. This project will also allow us to conduct an assignment as a pseudo-real research work.

## III. Past experience in ML

We had a few notions of machine learning. Let's sum it up. Charles had some fun recently discovering multivariate statistics (PCA, LDA, ...) but nothing that revelant nor very handy for this task. Even if it's closely related. Pierre had no past experience and Alexandre worked machine learning as a small part of a six months internship, experimenting with TensorFlow in an effort to train an artificial intelligence.

## IV. Our implementation

For our implementation, we choose to used Python as a language not because of its speed, memory consumption or runtime errors but because it was easy and fast for us to script and quickly test models while enjoying many machine learning frameworks. We decided to use scikit-learn because ~~it is Français~~ of its exhaustive documentation and its integration with other libraries like numpy, pandas and matplotlib. We indeed relied on those to manipulate and display data. From a technical point of view, we worked in test driven development especially while working on features to ensure a dependancy-less and long-lasting code but also to unitary test feature code and anticipate various kind of failure (some due to Python runtime politics (mostly dynamic typing and exceptions)). To ensure that the code will remain consistent and readable, we used pycodestyle, a PEP-8 compliant linter. We used Git as a version control system and Gitlab issues to discuss on the project.

### A. Gathering data

We have a python package data that is responsible for gathering, saving and loading data.

### B. Adding features

We began to add two simple features before having the whole loop. We such began with the length (as advised) and the subdomain count features.

### C. Features

We will describe some of our features and the reason the they might be interesting. Of course, we should limit strong preconceptions among their ability to classify properly URLs. We tried to find features that have the most unique characteristics and such the greatest fingerprinting capabilities, whether it be for safe or unsafe URLs.

*1) length:* Length may be a useful metric, shortest domain names are usually the most expensive, unless for new gTLDs.

*2) subdomains:* We can link an excessive amount of URLs to a particular domain thanks to the subdomain diversity. Subdomains are cheap to set up and manipulate. They can be used in a funny way with domain name (e.g. pay.pal.com ) which may lead to target misinterpretation.

*3) TLD usage:* Old domain names can not / generally not use new gTLDs. The .com popularity is nearly 50% whereas the new .dev is still less than 1%. [7] There is a large gap that might feed our model properly.

*4) days since registration of the domain:* Here we have to use network I/O which is pretty slow, but we think the nature of the data worth it. We think that unsafe domains are newly registered (to remain stealh or because older one had been detected).

*5) is ascii:* We are checking if the url charset can be restricted to ascii only. If not, it permits mono-letter homograph attack because homographs aren't bound to the ascii charset.

*6) russian TLD:* Adding the .ru russian TLD recognition as a feature surprisingly increased the model, what came up as a joke remained as an asset.

### D. Visualization

Visualization of data is a very important step on a data analysis approach. In our case, it helped us in feature selection (quality over quantity) and in model validation. We used two visualization tools in particular for feature selection:

*1) Scatter matrix:* The scatter matrix allows us to visualize the relationship between two variables. It's an estimation of a covariance matrix. The figure 1 shows how we should interpret the contents of such a matrix. The figure 2 is an exemple of perfect correlation between two variables (which are the same for the sake of the example). We are aiming for positive or negative linear relation between blue and red points. The more the line is balanced (diagonal in the square), the more strong the relation between the variables is. Scatter matrix is also a way to reveal cluster of points which can orientate us in the model of classification to be used and its parameters.

*2) Correlation matrix:* The correlation matrix is another tool to visualize the relationship between two variables. For that one, we linked all the variables in a triangle where the color hue of variable intersections expresses the correlation coefficient for the two variables intersected. The figure 3 is an example some of our features. As expected, we can see a strong correlation between the number of subdomains and the length of the URL.

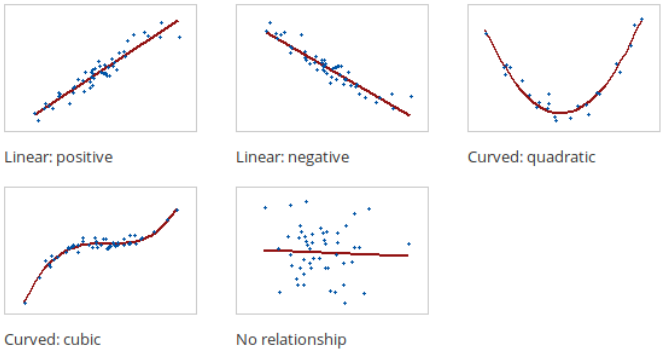


Fig. 1. Plots interpretation

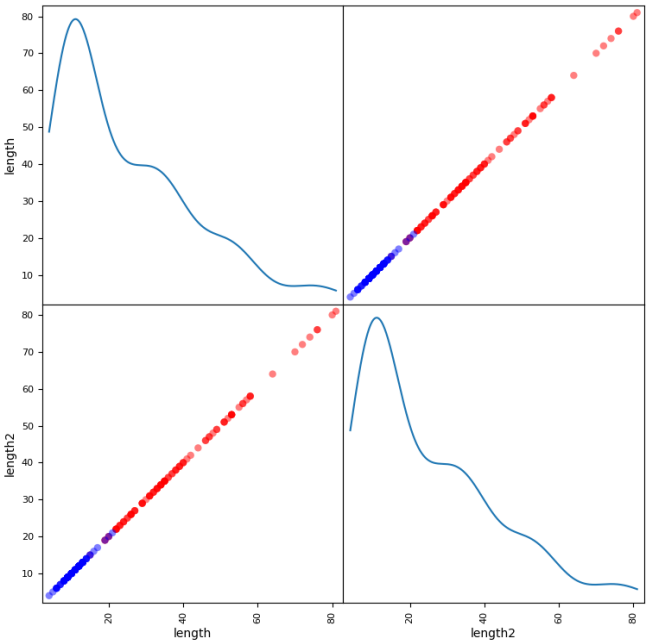


Fig. 2. Strongest relation possible

### E. Classifiers

During our first steps in the project, we tried to behave like a classifier, asking ourselves how should we proceed to classify a URL between two states : safe or unsafe. As a human we can easily observe independant characteristics of an object. For a URL, we observed the length and the number of subdomains at first glance. These are independant characteristics that suggests a correlation: the more subdomains a URL have, the more it is long. As a human we can treat length characteristic as a conditional variable depending on subdomains, in mathematics, such reasoning is attributed to Thomas Bayes (and its Bayes' theorem). We found during our researches that we can use this theorem to classify data, it is known as the naive bayes classifier. It's a linear classifier based on applying Bayes' theorem with strong (naive) independance between observed characteristics (features). As we continued ou researches, we found that this kind of frequentialist approach (or Bayesian inferential approach) was not the only one and we quickly felt overwhelmed by all the common techniques for supervised classification problems. We discovered SVMs (Support-Vector Machines) (another linear classifier), kNN technique (k-Nearest Neighbors algorithm), NN (Neural Network and derivates), decision trees and its agglomeration in random forets and some more. With only length and subdomain count as features, we were able to represent our training and testing datasets on a 2D graph, what we did. The figure 4 represents the different classifiers we tried over our dataset. The blue points

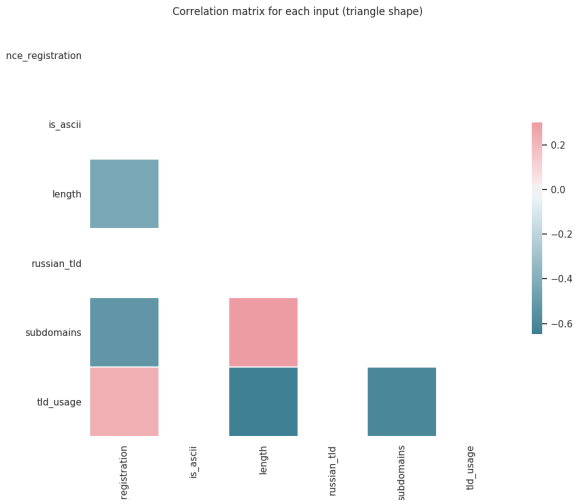


Fig. 3. Correlation matrix

corresponds to the unsafe URLs and the red ones to the safe ones. The semi-transparent points are the testing points and the plain ones, the training ones. The high accuracy is due to the fact that we use approximately 95% of 1-subdomain safe URLs: we can easily spot the red cluster aligned on the bottom line for one subdomain. Note that subdomains go from 1 to 3. However, it's interesting to visualize how these classifiers are splitting the 2D space. Visualization turned out to be very handy to detect problems with our datasets and to select the most appropriate features.

## V. Datasets

Over the time, we built two datasets for safe and unsafe URLs from various sources. Those datasets are dynamic and are updated whenever our sources are updating theirs. We support caching as well.

Unsafe URLs sources :

- PhishTank project api [4]
- CyberCrime database [5]
- University of New Brunswick [2] [3]

Safe URLs sources :

- Alexa top 1 million websites [1]
- University of New Brunswick [2] [3]

## VI. Our performance

The following results were obtained from a set of 1337 safe URLs and 1337 unsafe URLs. 60% of those have been used to train the models and the 40% remaining to test the models.

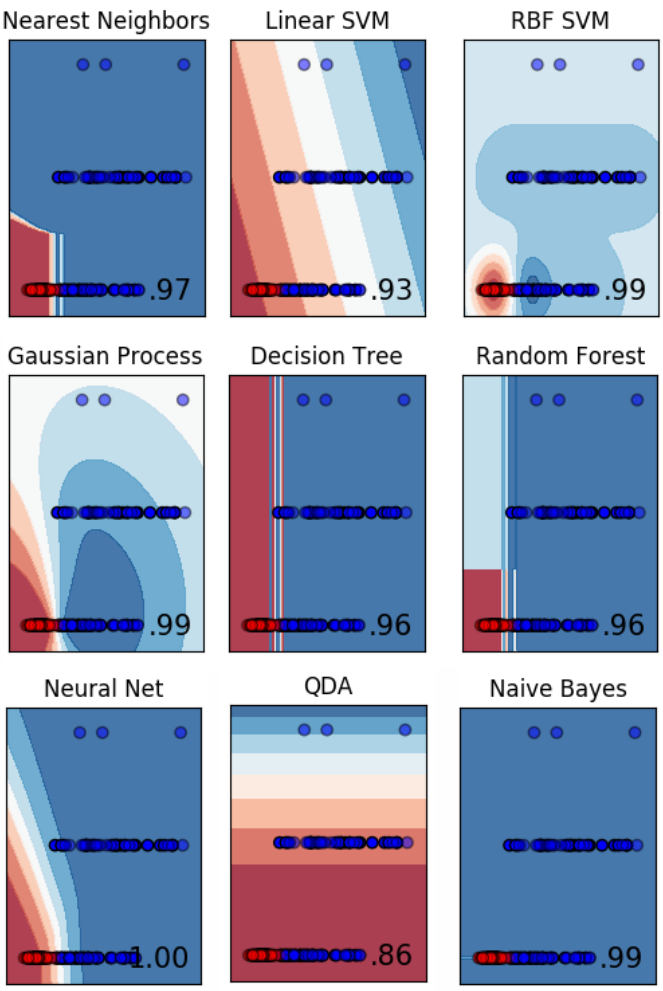


Fig. 4. Classifiers

| Classifier | Accuracy | Parameters |
|---|---|---|
| Linear SVM | 0.928 | penalty = 0.025 |
| RBF SVM | 0.953 | kernel coef = 2, penalty = 1 |
| Nearest neighbors | 0.919 | 3 nearest neighbors |
| Decision tree | 0.984 | m depth = 5 |
| Random forest | 0.974 | m depth = 5, n estimators = 1, f max = 1 |
| AdaBoost | 0.984 | |
| Naive Bayes | 0.983 | |
| LDA | 0.830 | |
| QDA | 0.981 | |
| Gaussian Process | 0.983 | kernel = 1.0 * RBF(1.0) |
| Neural Network | 0.941 | |

The timings (with data reading / writing steps) :

- 579,09s user
- 285,23s system
- 46% cpu
- 30:42,48 total

## VII. Conclusion and future work

We enjoyed the project as we worked on new topics and discovered some cool things around multivariate statistics, data mining and analysis, data visualization, classification and prediction. Our strongest correlation will remain the one linking the length and the number of subdomains of a URL.
A first easy step to implement would be adding a validation test to do model selection [6]
One big improvment step would be to build I/O to the program in order to allow new URLs. We could then proceeed to unsupervised learning. We could connect this I/O system to a fancy front-end in order for people to test their URLs while increasing the accuracy of the model but we can also link it to mail server in order to classify and prevent malicious mails, or any live stream of data that would help increasing the model relevance. Another good step would be to dynamically test features (may it be bucket or split testing) in order to be able to choose them more specifically and to reveal possible local maxima.

Here are some pertinent future feature ideas:

- Pagerank computation (api limitations)
- Prohibited statuses in whois infos (registrant takes care of its domain)
- Count redirections (is website stable ?)
- Homoglyphs and the internation domain name (IDN) homograph attack
- Typosquatting (can compare Hamming distance to safe brand names)
- IP blacklisting (can use passive DNS)

## References


[1] Amazon. Alexa top 1 million websites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[2] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani. University of new brunswick – canadian institute for cybersecurity. https://www.unb.ca/cic/datasets/url-2016.html.

[3] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani. Detecting malicious urls using lexical analysis. pages 467–482, 2016.

[4] PhishTank. Phishtank database. http://data.phishtank.com/data/online-valid.csv, https://data.phishtank.com/developer_info.php.

[5] C. Tracker. Cybercrime full list. http://cybercrime-tracker.net/all.php.

[6] S. University. Model selection and train/validation/test sets. https://www.coursera.org/lecture/machine-learning/model-selection-and-train-validation-test-sets-QGKbr.

[7] a. d. o. Q.-S. W.-b. S. W3Techs. Usage of top level domains for websites. https://w3techs.com/technologies/overview/top_level_domain/all.