SD201 - Project
Phishing Website Detection

Guillaume Paya-Monet
Khalil Braham
Souheib Ben Mabrouk

SD 201
2022-2023

Contents

1 Introduction **4**

2 Building the Dataset **4**
2.1 Data Collection 4
2.2 Feature Extraction 5
2.3 Computing Features for the Data 7
2.4 Data Pre-processing and Visualisation 7
2.4.1 Merging all the data 7
2.4.2 Dealing with non numerical data 7
2.4.3 Dealing with missing data 7
2.4.4 Dealing with outliers 8
2.4.5 Transforming the features 9
2.4.6 Data correlations 9

3 Models Selection **10**
3.1 Decision Tree 10
3.2 Random Forest 10
3.3 K-NearestNeighbor 10
3.4 Naive Bayes 11
3.5 Support Vector Classifier (SVC) 12
3.6 Multi-layer Perceptron (MLP): Deep Learning 12

4 Evaluation of the Models and tuning of parameters **12**

5 Conclusion **13**

6 Who did What? **14**

List of Figures

1 Evolution of phishing attacks and trend in 2020[1] 4
2 PhithTank logo and Wepscraping explanation 5
3 Structure of an URL link 5
4 Visualisation of the domain length feature distribution and boxplot before eliminating the outliers 8
5 Visualisation of the domain length feature distribution and boxplot after eliminating the outliers 8
6 occurrence_tirets initial countplot 9
7 occurrence_tirets final countplot 9
8 Correlation matrix 10
9 Evolution of accuracy against the hyperparameter k 11
10 Feature importance and ROC-curve for Random Forest and decision trees 12

1 Introduction

Malicious websites are everywhere on the internet. As our world becomes increasingly digital, more and more of our personal data is stored and manipulated on the web servers. For a normal person, this data can be private, intimate, whether it is on social media, on a bank account or even an email account, it has an immense value for each and every human being. That's why some individuals will go to lengths to steal this data either to sell it to a higher bidder or to ask you for ransom or even for blackmailing purposes.

Figure 1: Evolution of phishing attacks and trend in 2020[1]

The main way these scammers will try to steal this data is with phishing websites. These are normal, legit looking websites, but with specific algorithms that tend to store the information we put into and send it to the scammers server instead of the legit website's server. This type of website can also have malicious code hidden in the HTML that can penetrate through a computer by accessing the IP address, and taking control of the system, almost unnoticeably. These scammers are usually professionals and know exactly what they are doing, so they make it very difficult for a normal person to distinguish their phishing website from the legitimate one.

Therefore, it is becoming more essential and more urgent than ever for individuals and organizations to be able to identify whether a website is safe to use or not. Thanks to data science and carefully chosen features, we are able to assess its safety before we use it.

Hence, this project has the goal to create a system that can generate and analyze different features of a given website and then identify, using Machine Learning its legitimacy.

2 Building the Dataset

2.1 Data Collection

The first step in our project is the collection of data and we chose not to use an existing dataset considering the continuous changes in phishing techniques and the diversity of malware design and architecture so we decided to go for web scraping. We wrote a python script using the **Selenium** library that we ran on **phishtank.org** to collect the most recent phishes.

PhishTank is an anti-phish website made by the company OpenDNS. Its methodology consists of giving the user an access to a big database of phishing websites. Then, the user can report suspected phishes and then a validation system validates whether or not it is a phish. [2]
Scraping the websites directly from **phishtank.org** makes us sure that our database is fresh for exploitation.

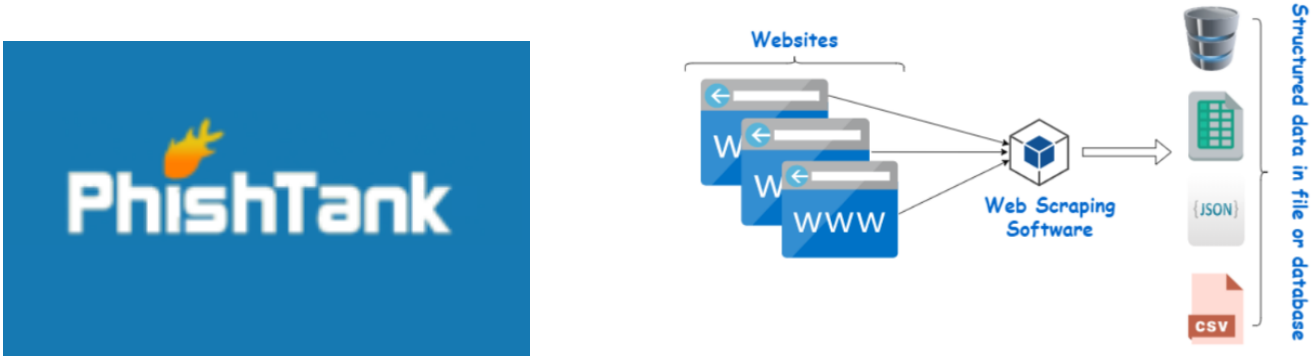

Figure 2: PhithTank logo and Wepscraping explanation
[3]

As for the legitimate websites, since May 2022, there have been no international references for valid links after the shutdown of the alexa amazon database. Hence, it seemed reasonable for to look for the top used websites in the world, so we decided to scrap a list of websites from **domcop.com/top-10-million-websites** using both selenium and BeautifulSoup.

We remarked that the websites we scraped tend to have a sort of a pattern. Hence, to avoid some kind of bias, we decided to extract a part of the dataset from other external sources such as kaggle.

We forced data balance from the beginning by selecting: **3000** Phishing websites and **3000** Legitimate websites to generate their features in the next step.

2.2 Feature Extraction

One of the most crucial parts of our project was to do a precise and efficient feature extraction in order to build significant data that can answer the problematic. To pick the features we were inspired by the paper "Phishing Websites Features" [4] .

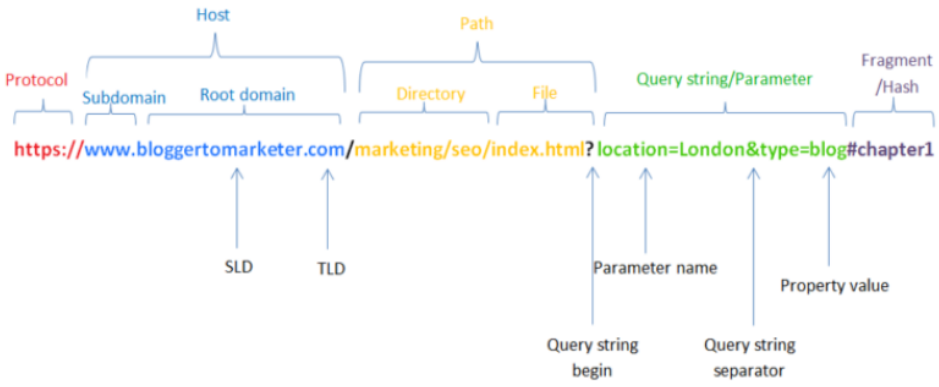

Figure 3: Structure of an URL link

The most challenging part of this step was to examine the state of the art methods and use the ones we found meaningful to create our features using different python libraries designed for cybersecurity applications. We wrote 18 functions to compute 18 different features.

1. **URL based Features:**

   1.1 Domain of the URL: We are just extracting the domain present in the URL. This feature does not have much significance in the training but we are using it to generate other features. Then, it will be dropped before training the models.
   1.2 IP address: We checked if the Url provided has a common IP address. Normally, legitimate websites have domain names and addresses in the subdomain. Some phish websites lose their domain once they are reported or stopped. This feature will also be used to generate other features later.
   1.3 Presence of @: When phishers include the @ character in a URL, the browser ignores anything before it, and the true address is commonly found after it. As a consequence, a website having @ character in its domain is more likely to be phishing.
   1.4 Length of URL: Phishers can use a long URL to hide the doubtful part in the address bar. In some references [5], we found out that if the length of the URL is greater than or equal 50 characters then the URL is more likely to be phishing.
   1.5 Length of Domain: Usually the domain of real websites does not exceed 20 characters.
   1.6 Depth of URL: The slashes '/' in Url indicate the directories. An increasing number in the use of slash is suspicious.
   1.7 Number of hyphens ('-'): The dash symbol is seldom used in official websites. Actually, many attackers use it to make you feel like you are addressing the real website like : paypal.com and pay-pal.com.
   1.8 Redirection: If the URL route contains the character "//," the user will generally be redirected to another website.
   1.9 Number of percent ('%'): It's a sign of URL encoding. A website having @ character in its domain is more likely to be phishing.
   1.10 https in Domain name: The phishers may add the "HTTPS" token to the domain part of a URL in order to trick users. For example, http://https-www-paypal-it.com/.
   1.11 Using URL Shortening Services "TinyURL": URL shortening is a method on the "World Wide Web" in which a URL may be made considerably smaller in length and still lead to the required webpage. It will be used by attackers to make a domain name that is short links to a webpage that has a long URL

2. **HTML and Java based features:**
   In this section, we will extract some information based on the HTML and JavaScript features of websites.
   2.1 iframe tag: IFrame is an HTML tag used to display an external webpage into the one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible.
   2.2 Status Bar Customization: Scammers may use JavaScript to show a fake URL in the status bar as a camouflage technique. To extract this feature, we must dig into the webpage source code, particularly the "onMouseOver " event, and check if it makes any changes on the status bar.
   2.3 Number of Redirects: This feature looks for how often a website redirects to another page before stopping when accessing the particular link. A legitimate website would redirect at most once.

3. **Abnormal Based Features:**
   3.1 URL of anchor: This consists in scraping the HTML of the code, iterating through the

<a> anchors and seeing if it links to any other domain. The ratio of abnormal anchor tags to total tags should be less than 31

4. **Domain Based Features:**
   4.1 WHOIS database: This feature checks if the link exists in the WHOIS database and calculates the number of months since the domain was added to this database. A domain created less than 6 months ago could be considered suspicious.
   4.2 Alexa 1M database: The Alexa Internet database holds a 1 million-long list of websites with the most traffic. If a website is not on that list, it could be suspicious.
   4.3 Number of links to domain: This feature looks at all the links in a web page and counts the number of links that redirect back to the domain. A legitimate website would most likely have at least 2 links.
   4.4 Google index: This feature checks whether a URL is indexed by Google. A website that isn't could be considered to be suspicious.

2.3 Computing Features for the Data

This part of the project was the most challenging for us. Collecting the features for the data took a long time because of the large number of URL links we decided to inspect and of the large number of features, the complexity of their extraction techniques and the abnormal scripts running the phishing websites.

Due to these types of scripts, some algorithms tend to fall into a seemingly endless loop, and some other algorithms could take more than 5 minutes for the extraction and still not return any valuable data. Hence to construct the whole dataset, we estimated a running time of 150 hours.

Due to time constraints, we decided to reduce the phishing and legitimate database size to around 3000 links each and we added a time constraint on every algorithm and to divide the dataset into parts where each one of us ran the algorithms on 2000 links.

These techniques enabled us to capture the 6000 URL links and the necessary features.

2.4 Data Pre-processing and Visualisation

This part is so important in our project since we designed the database from scratch. So we had to deal with unusually dirty data. We will present our principal techniques but all the details are explained in the notebook for every feature.

2.4.1 Merging all the data

We had to do multiple runs on data from different sources to generate the features, so the first part was to merge and add all the parts of the dataset in one final version.

2.4.2 Dealing with non numerical data

The dataset contained a set of non numeric features, such as Link and Domain. After analyzing their values, we found out that they contain unique values so they are considered as indexes. Consequently we dropped them.

2.4.3 Dealing with missing data

The second part was to deal with the anomalies that we found in the dataset. We had an important amount of None values everywhere due to time surpassing reasons or to the nature of

web scraping algorithms.

However, the data is not as bad as it seems, we did not need to drop these features because these Nan values were not really missing, but had a significance. We decided to iterate through the features, visualize the data distribution and values and interpret the None values depending on the scraping algorithm and the values distribution.

In most of the features, we replaced the None value with 0 : This technique was useful when a None value was added after a searching function surpassed a threshold or in an infinite loop. Since Beautiful soup won't return an error, the None value reflects a non presence of the value, or even the non presence of the website that we were supposed to search into which is reasonable especially in the phishing websites that have been removed or hidden. Hence, the None value represented by 0 in these features : checkWHOIS_exists, NumbLinktoDomain, NumberRedirects, isinAlexa, IndexbyGoogle, MouseOver.

2.4.4 Dealing with outliers

Going through the features, one by one, we visualized the distribution and the boxplot of each one of them. In some cases, we had many outliers that we had to deal with. In most of the cases, these outliers were added through a counting algorithm. They usually represent a part of the (Q3) quartile. Hence, we decided to give them a maximum value that we chose based on the initial distribution and the meaning of the feature.

As an example, we take the Domain Length feature :

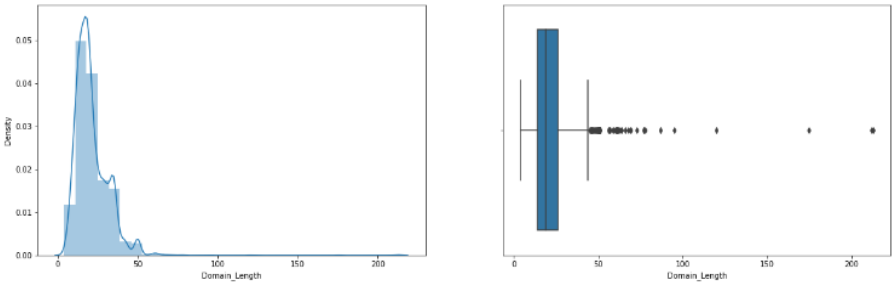

Figure 4: Visualisation of the domain length feature distribution and boxplot before eliminating the outliers

Here we noticed that the initial distribution showed a trimodal function with 3 maxima and with outliers going up from 50, we had to impose a threshold and give them a value of 50 to keep the importance of the (Q3) quartile without having to consider them as outliers.

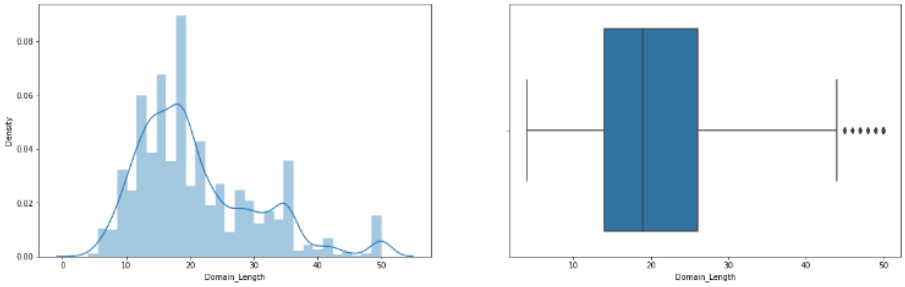

Figure 5: Visualisation of the domain length feature distribution and boxplot after eliminating the outliers

2.4.5 Transforming the features

In some of the features, we decided to interfere in the values as we remarqued a certain pattern in the values distribution that we were able to highlight.

For example, we take the occurrence_tirets feature. By analyzing the countplot, we saw that when the number is equal to one, the label is more likely to be 0. Starting from 2, the result is the opposite with many outliers starting from 3.

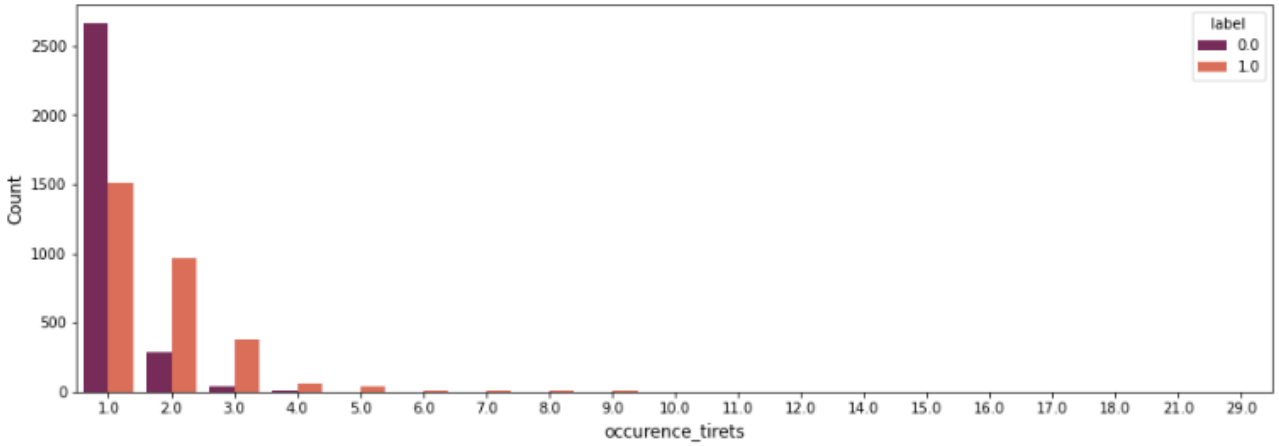

Figure 6: occurrence_tirets initial countplot

As a consequence, we took the decision to transform the feature result to :

- 1 if the number of occurrences is 1
- 0 if it's higher than one.

As result, we had this distribution that we found this distribution that seems to be interesting as it divides the values into two classes where we clearly see a correlation with the result:

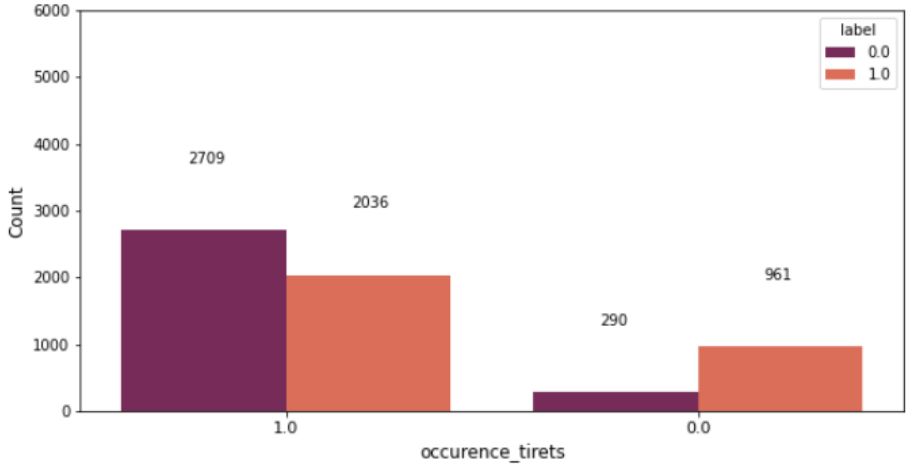

Figure 7: occurrence_tirets final countplot

2.4.6 Data correlations

As we can see, the correlation heatmap does not show a remarkable correlation between features, but we have some relations that seem reasonable such as Domain Length and URL Length,

Redirections and URL Length. But since we are dealing with a low dimensional dataset after cleaning, we decided to keep all the left features.

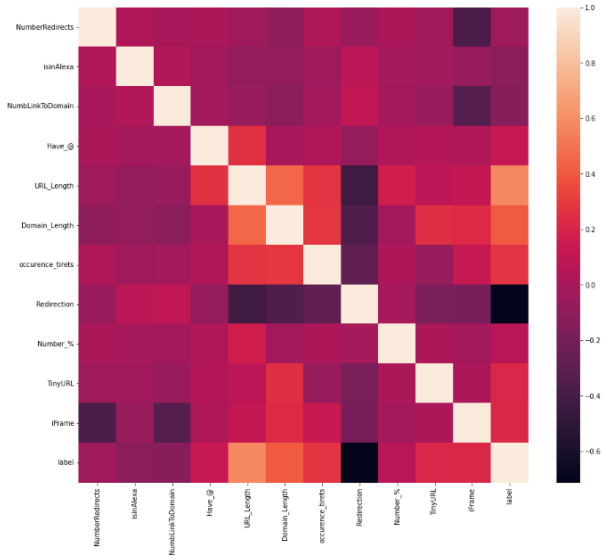

Figure 8: Correlation matrix

3 Models Selection

3.1 Decision Tree

A Decision Tree is a type of supervised machine learning model where the data is continuously split according to a certain parameter called Gini. This parameter quantifies the uncertainty about the value of the target feature. It is calculated by subtracting the sum of the squared probabilities of each class from one.

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2$$

This lets us choose the condition on the features in each node that reduces the uncertainty in order to divide the dataset.

3.2 Random Forest

Seeing that random forests combine the simplicity of decision trees with flexibility to new data input, we chose to use this model for a potential improvement of the prediction accuracy of our model. data input, we chose to use this model for a potential improvement of the prediction accuracy of our model. Random forest is a supervised machine learning algorithm widely used in classification that builds decision trees on different samples and takes their majority vote.

3.3 K-NearestNeighbor

This supervised model tries to predict which classes an object belongs to by looking at K of the nearest neighbors of that object and selecting the class which is most prominent amongst that list

of neighbors of size K. The distance used measured between two points is the Euclidean distance: $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.

We can plot the accuracy of the prediction on the test data against the hyperparameter K representing the number of nearest neighbors we pick:

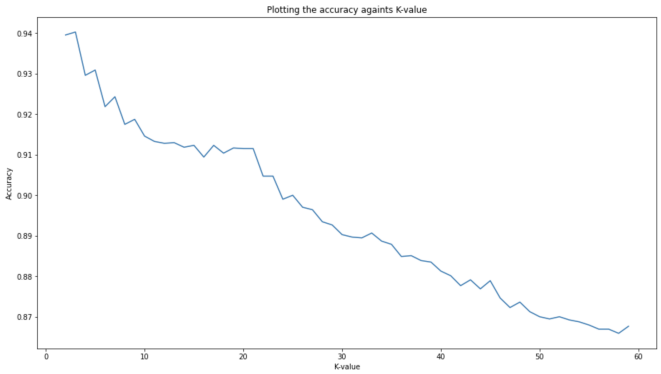

Figure 9: Evolution of accuracy against the hyperparameter k

We can notice that after $K = 20$, we reach overfitting which occurs because the classifier predicts a bit too well on the training set and then doesn't handle the test dataset, giving more faulty predictions.

We can choose here $K = 3$ which gives the best accuracy and doesn't lead to overfitting or underfitting.

3.4 Naive Bayes

This supervised probabilistic classifier is based on applying Bayes' theorem $P(x|y)$ where we try to measure the probability of an object belonging to a class considering observations (the features). We then get for the classifier, $P(C_k|x_1,...,x_n)$, where for each k class (in our case either phishing or legit), we calculate the probability of being in that class depending on n features.

To model the events, we can use multiple assumptions that the classes are distributed depending on a specific distribution. Different distributions will then generate different accuracy. We have:

- **Gaussian** Naive Bayes: based on Normal distribution
- **Multinomial** Naive Bayes: a generalization of binomial distribution
- **Bernoulli** Naive Bayes: based on Bernoulli distribution
- **Complement** Naive Bayes: adaptation of Multinomial Naive Bayes to handle imbalance dataset

We can then see which distribution has the highest accuracy:

- Gaussian NB: 81% accuracy
- Multinomial NB: 70% accuracy
- Bernoulli NB: 70% accuracy
- Complement NB: 73% accuracy

Here the Gaussian model predicts with the highest accuracy.

3.5 Support Vector Classifier (SVC)

The Support Vector Classifier tries to find the best hyperplane in order to separate the different classes. Then when a point belongs to a particular hyperplane, it means that it belongs to the particular class that represents that hyperplane.

We can use the SVC classifier using multiple kernels, linear or polynomial using different degrees:

- Linear Kernel: 89% accuracy
- Polynomial degree 2 Kernel: 85% accuracy
- Polynimial degree 3 Kernel: 84% accuracy
- Polynomial degree 4 Kernel: 69% accuracy
- Polynimial degree 5 Kernel: 66% accuracy

The SVC kernel with the best accuracy is the linear kernel.

3.6 Multi-layer Perceptron (MLP): Deep Learning

We use the MLPClassifier class that implements a multilayer perceptron algorithm that trains by backpropagation for supervised learning tasks. MLPs can be thought of as generalizations of linear models that perform several processing steps to reach a decision. This classifier is most likely to be used when the data is not necessarily linearly separable.

4 Evaluation of the Models and tuning of parameters

Since we have an equal number of samples belonging to each class we choose to work with accuracy metric as the ratio of number of correct predictions to the total number of input samples:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

TP : True Positives | **FP** : False Positives | **TN** : True Negatives | **FN** : False Negatives

For some models we tried to highlight the features' importance to do the classification and the ROC curve as the example below.

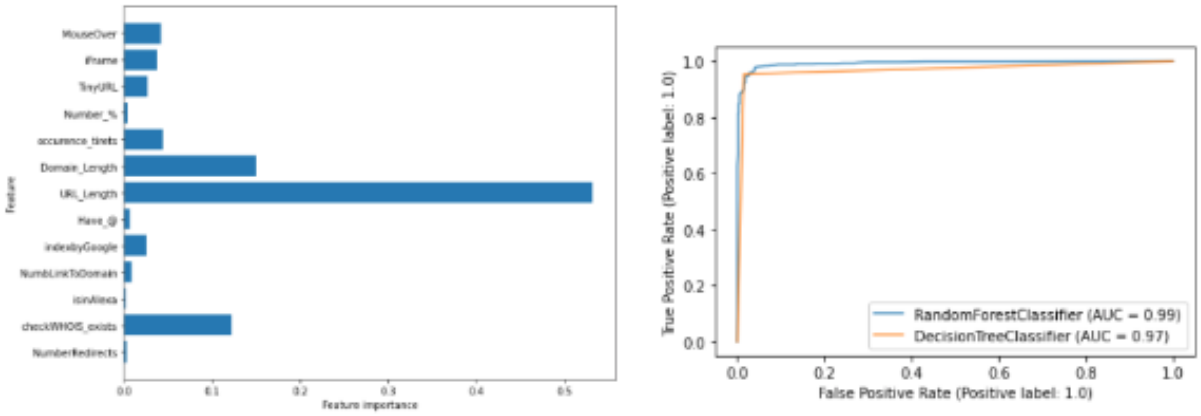

Figure 10: Feature importance and ROC-curve for Random Forest and decision trees

For each model, we ran a script taking an interval for each parameter. Using RandomizedSearchCV, the algorithm randomly takes a value from each interval and runs the model and then compares the result with other runs based on a target score. This method allows us to select the best hyperparameters for each model.

Model	**Accuracy**	**Optimal parameters**
Decision Tree	0.919	min samples leaf: 2, max depth: 7, criterion: gini
Random Forest	0.921	n estimators: 500, max features: 'sqrt', max depth: 8, criterion: entropy
KNN	0.94	K=3
Naive Bayes	0.81 (Gaussian NB)	var smoothing: $1e-08$
SVC	0.89 (Linear Kernel)	kernel: linear, gamma: 0.001, C: 1
MLP	0.988	solver: lbfgs, random state: 3 hidden layer sizes: 12, alpha: 0.0001

An analysis of the results shows a maximum accuracy of 98% with the **MLP** approach. However, The MLP being a costly approach, it will require lots of computational power to keep the model up to date with an ever increasing list of phishing websites. Therefore the next model can be chosen, the **K-Nearest Neighbor** model, with a 94% accuracy which still allows us to correctly predict the status of a website with a very respectable precision. With **KNN** no training is required before making predictions, therefore new data can be added seamlessly and would be overall the best approach.

5 Conclusion

This project has been a very enriching experience, as we had the chance to explore every step of a machine learning pipeline. Starting from the problem analysis, to formalizing it into a data mining project, to gathering the data from different sources using many state of art methods to extract significant features, then going through, visualizing and cleaning a realistic and imperfect data to finally implementing and optimizing various models to have our final results.
We encountered many problems such as choosing the right features by reverse engineering the problem, hardware limitations during the features extraction and handling missing values due to the nature of phishing websites.
We are happy with the final version of our project, as it uniqueness consists of our business understanding. We started from nothing but a deep research into the technical details of the problem. Based on this acquired knowledge, we built our precise features that go directly to the point, hence our good precision results.
However, we think we could have had a better result if we had less missing values, more features and a bigger dataset. Also, we would have preferred if we had a final product with an API to make our service open for the users and complete the Machine Learning pipeline. This could have been reached by having a specialist opinion for more features, using our GPUs and parallel computing for faster and more efficient feature extraction.

Finally, we came to the conclusion that every data scientist should cherish his data engineer, as we lived the role and we powered through its challenges.

6 Who did What?

Since the early steps of the project, because of the complexity of the algorithms and their long execution time, we had to work together as a group and divide each task so we could tackle it efficiently.
As the first step went well, kept the same energy throughout the rest of the project so we did every part together until this report.

Khalil Braham : Web scraping, Data construction, Data cleaning and visualisation, parameters tuning.
Souheib Ben Mabrouk : Web scraping from phishtank, Construction of feature extraction functions, Phishing-websites data construction, Decision Tree model, Random Forest Model, MLP model.
Guillaume Paya-Monet: Web scraping, Construction of feature extraction functions, Phishwebsites and legit data construction, KNN model, Naive Bayes Model and SVC Model.

References

[1] Savernova. Top 5 trends in email security for 2021. [**CrossRef**].

[2] Wikipedia. Phishtank. [**CrossRef**].

[3] Adib Habbou. Introduction à selenium : Web scraping avec python. [**CrossRef**].

[4] Lee McCluskey Rami M. Mohammad, Fadi Thabtah. Phishing websites features. [**CrossRef**].

[5] Sapra Aman Kumar Naman Kumar S Arthi Akashdeep Bhardwaj, Varun. Why is phishing still successful? [**CrossRef**].