PROJECT REPORT

(Project Semester January 2018 to June 2018)

noPhish - a Chrome Extension to classify websites into Phishing and Legitimate (Non - Phishing) using Machine Learning, Image Processing, and Data Mining techniques.

Submitted by

Yash Saboo

15070121170

Under the Guidance of

Faculty Mentor

Mrs Prachi Nitin Kadam

Assistant Professor

Industry Mentor

Mr Akash Verma

Software Engineer

Department of Computer Science and Information Technology

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

January 2018 to June 2018

DECLARATION

I hereby declare that the project work entitled "noPhish - a Chrome Extension to classify websites into Phishing and Legitimate (Non - Phishing) using Machine Learning, Image Processing, and Data Mining techniques." is an authentic record of my own work carried out at Symantec Software India Pvt. Ltd. as requirements of six months project semester for the award of degree of B. Tech. (Computer Science), Symbiosis Institute of Technology, Pune, under the guidance of Mr Akash Verma and Mrs Prachi Nitin Kadam, during January to June 2018.

Yash Saboo

Date: 18 June 2018

15070121170

Certified that the above statement made by the student is correct to the best of our knowledge and belief.

Mrs Prachi Nitin Kadam	Mr Akash Verma
Assistant Professor	Software Engineer
Faculty Mentor	**Industry Mentor**

Symantec

June 18, 2018

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Yash Saboo (Emp ID – 282048) was undertaking an internship in our organization and has completed the project. Details of the same are provided below:

Start Date	:	*January 10, 2018*
End Date	:	*June 18, 2018*
Title of the project	:	*Develop a Chrome Extension to classify websites into Phishing and Legitimate (Non - Phishing) using Machine Learning, Image Processing, and Data Mining techniques.*
Name of Project Leader	:	*Sainarayan Nambiar*

Sincerely,
For Symantec Software India Pvt. Ltd.

Sudhanshu Pandit
Vice President - Human Resources

ACKNOWLEDGEMENTS

I thank Training and Placement Cell of SIT Pune for arranging this internship opportunity, and hosting the interviews meticulously.

I thank Symantec for providing an opportunity to work on a research project of great significance to the core competence of the company, with a multinational team of truly talented coders, and access to the best computing and networking infrastructure and resources.

I am grateful to my Faculty Mentor, Mrs Prachi Nitin Kadam, and Industry Mentor, Mr Akash Verma, for extending patient guidance, advice, and opportunities to improve upon my work.

I also express my sincerest thanks to the other interns, engineers, managers, and faculty members whose cooperation made this internship very fruitful, enjoyable, and memorable.

Contents

Article I. Abstract

Phishing is the attempt to obtain sensitive information such as usernames, passwords, and credit card details, often for malicious reasons, by disguising as a trustworthy entity in an electronic communication. Major browser vendors race through hundreds of thousands of websites a day, assessing each URL to find Phishing websites to add to their blacklists. The quantity of suspect sites is so large that automated classification is essential.

Creators of Phishing sites are aware of the popular anti-phishing techniques and implement a variety of countermeasures to disguise the site from software while remaining recognisable to humans. Consequently, researchers are exploring machine learning and computer vision to better recognise Phishing sites.

I present noPhish - a novel approach to identify the targets of potential Phishing attacks using two-level validation. The first level of noPhish is a XGBoost machine learning model which predicts if the website is Phishing or not, based on technical features of the website. The second level of noPhish combines computer vision techniques with density-based clustering to identify a phishing website by its visual aspects.

The two-level validation helps in lowering false positive. noPhish will be integrated with Norton web extension for end-users. The extension will work in real-time and classifies a website as phishing or safe.

Article II. Introduction

Why does phishing work? Basically because con artists are really good at persuading people to do really dumb things.

– Richard Clayton, Cambridge Computer Laboratory

A career as a cyber-criminal has never been so attractive. The pay-outs are high, convictions are difficult, and the required technical expertise to get started is surprisingly low. Cyber-criminals exploit the fully globalised nature of the Internet and enjoy the protection offered by a fragmented global justice system and poor extradition policies.

One of the easiest ways to start one's career in cyber-crime is Phishing. Phishing is a form of online fraud, where criminals exploit the implicit trust placed in an organisation's branding to acquire sensitive information from the organisation's customers. This sensitive information can be used for identity theft, bank fraud, or as part of a larger campaign to compromise the victim, their employer, or their national government.

The World Wide Web is a popular platform for Phishing. A Phishing website is a website that masquerades as a trusted brand, but is actually controlled by a criminal. All data sent to the website, such as user names, passwords, and credit card numbers, can then be utilised by the criminal or sold on the black market.

The Internet community has responded to the threat users' face from Phishing. All major desktop and mobile web browsers (Microsoft Internet Explorer, Microsoft Edge, Google Chrome, Mozilla Firefox, Apple Safari, Opera) use blacklists to display warnings to their users when they attempt to access a known Phishing site.

It is fundamental to the effectiveness of Phishing blacklists that they are updated regularly. Phishing sites are created at a rate of over 70,000 websites a month. The scale of Phishing activity has led to the automation of classification of Phishing websites. Due to the widespread use of the web browsers (Google Chrome has over a billion users), the impact of the blacklists is massive, and, transitively, so is the impact of the classification process. High precision is essential.

Article III. Motivation

The most basic aspect of a phishing website is that it needs to look similar to the actual website for deceiving a user. Phishers try to imitate the original webpage as much as they can, keeping the authenticity (technicality) intact. So, if an anti-phishing technique can be developed which predicts in real-time if a website is phishing not only based on its technicality, but also on its visual appearance. 'Real-time', here, refers that the client should be kept protected from phishing websites during its regular browsing activity.

The level 1 of noPhish tries to predict if the website is phishing by using the technical aspects of the website. The 'technical aspects' refers to various features that helps in distinguishing phishing websites from legitimate websites such as domain life, verified SSL authorizing agent, JavaScript events, page links, HTTP Headers and many more such features. For each candidate website, extract multiple features of the page and test it on the XGBoost machine learning classifier which has been trained on a huge pool of websites, which consisted of both legitimate and phishing websites.

The level 2 of noPhish tries to weigh importance only on the visual appearance of the website. If the page didn't look like a trusted brand the user wouldn't be fooled into submitting their sensitive information. We can make this an image classification problem. For each candidate page, take a screenshot of the page and compare its similarity to screenshots of known legitimate sites. If the candidate looks similar to one of the legitimate pages, and the URL of the candidate does not match, we can consider it Phishing.

Article IV. Work Industry Review

Symantec Corporation (NASDAQ: SYMC), the world's leading cyber security company, allows organizations, governments, and people to secure their most important data wherever it lives. Enterprises across the world rely on Symantec for integrated cyber defence against sophisticated attacks across endpoints, infrastructure, and cloud.

More than 50 million people and families rely on Symantec's Norton and LifeLock Digital Safety Platform to help protect their personal information, devices, home networks, and identities at home and across their devices. Headquartered in Mountain View, Calif., Symantec has operations in more than 40 countries.

Section 1.01 Software Portfolio

- **Enterprise Security**

  Symantec protects the Cloud Generation through an Integrated Cyber Defense Platform, the industry's most complete portfolio for securing cloud and on-premises environments. They support 15,000 enterprises in taking full advantage of cloud computing without compromising the security of the people, data, applications, and infrastructure that drive their business. Their advanced technology portfolio is powered by the world's largest civilian threat intelligence network, enabling them to see and protect against the most advanced threats.

- **Consumer Security**

  The solutions and subscriptions from Symantec's consumer brands, Norton and LifeLock, comprise the Digital Safety platform, which protects information, devices, networks, and identities. Norton-branded solutions provide multi-layer security for desktops, mobile operating systems, and home networks, defending against increasingly complex online threats to individuals, families, and small businesses.

Section 1.02 Points Worthy of Note

- Norton is the market share leader of paid Consumer Endpoint Security.
- Norton Security won AV-Test's annual 'Best Protection' Award in 2015 and 2016.
- Symantec Advanced Threat Protection 100% effective against advanced techniques in Dennis Technology Labs 2015 report.
- LifeLock received Top 10 placement on Online Trust Alliance (OTA) Audit Honor Roll.

Section 1.03 What Sets Symantec Apart?

Symantec's teams around the world are developing technologies and building solutions to help customers secure and manage their information. The company has a robust portfolio and a long history of technology leadership.

- **Security Technology and Response (STAR)**
  Symantec's Security Technology and Response (STAR) division, which includes Security Response, is a global team of security engineers, virus hunters, threat analysts, and researchers that provides the underlying security technology, content, and support for all Symantec corporate and consumer security products (2).
- **Symantec Research Labs (SRL)**
  Symantec Research Labs (SRL) is Symantec's global research organization and has played a leading role in developing and commercializing numerous cutting-edge technologies across Symantec's business areas. Commercialized technologies from the group include industry leading rootkit protection, innovative browser protection technology to proactively block future exploits of known vulnerabilities, Symantec's first antispam technology.

I was a Research Undergraduate Intern in STAR team at Symantec, Pune**.**

Article V. Background

Section 1.04 Phishing

A Phishing site, Phishing page or - colloquially - Phish is a website that seeks to mislead a victim into giving sensitive information to the site's controller - a Phisher. Phishing websites are designed to be similar to their target - a legitimate website, often a financial institution or social network, where the credentials have value to the Phisher either personally on the black market.

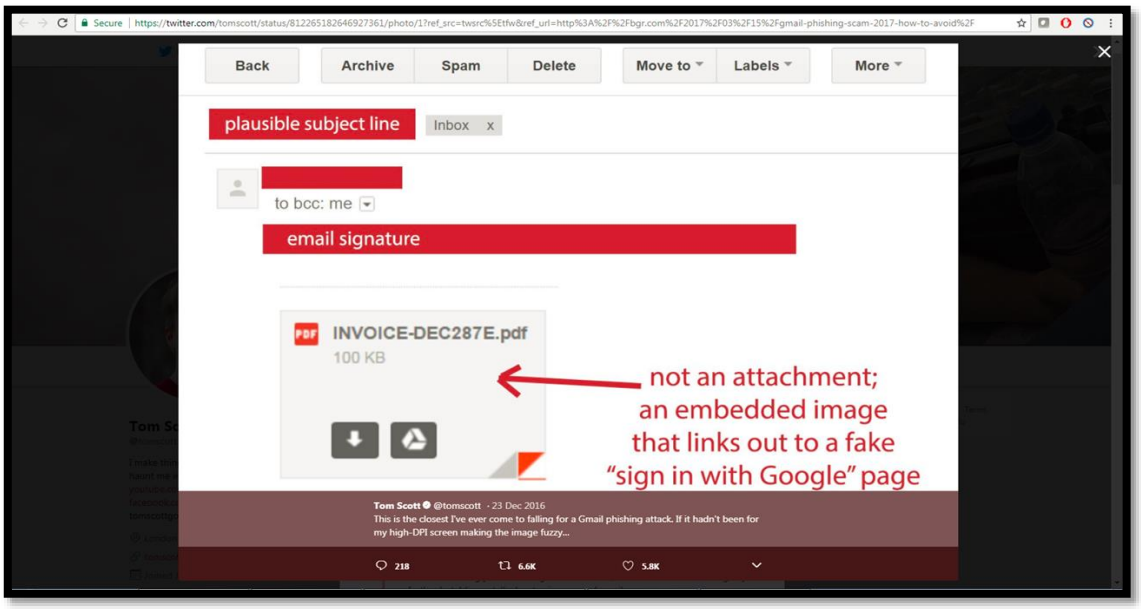


Figure 1. The picture shows a clever way phishers use to trick users in emails.

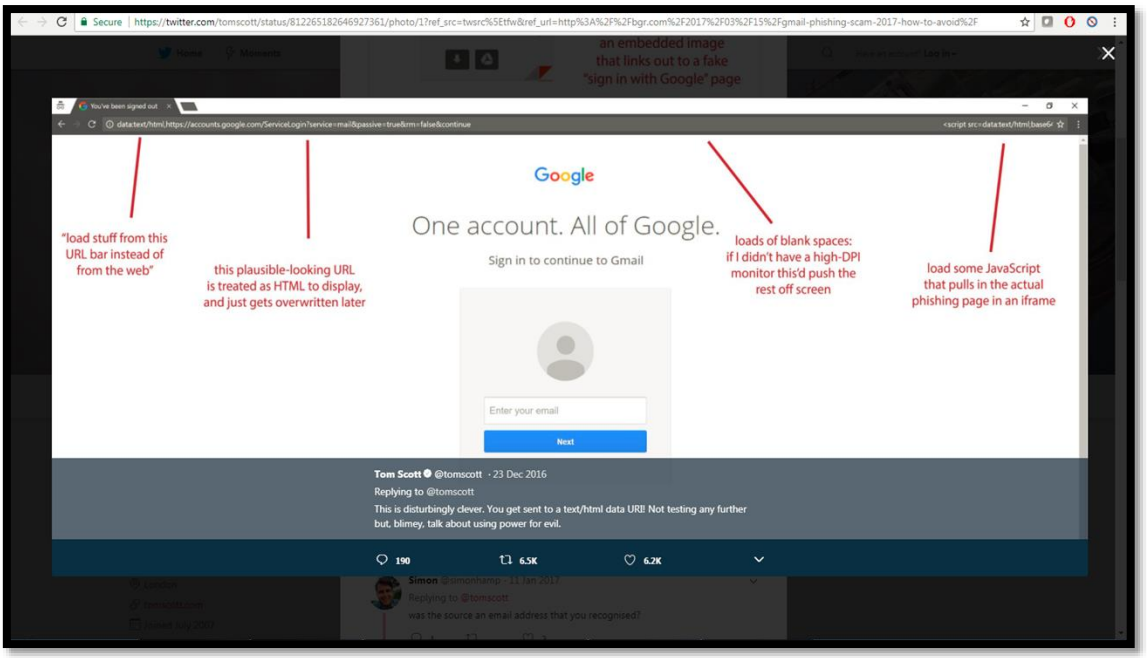


Figure 2. A tweet by Tom Scott which explains an ingenious way used to phish Google users.

Section 1.05 How Phishing Works?

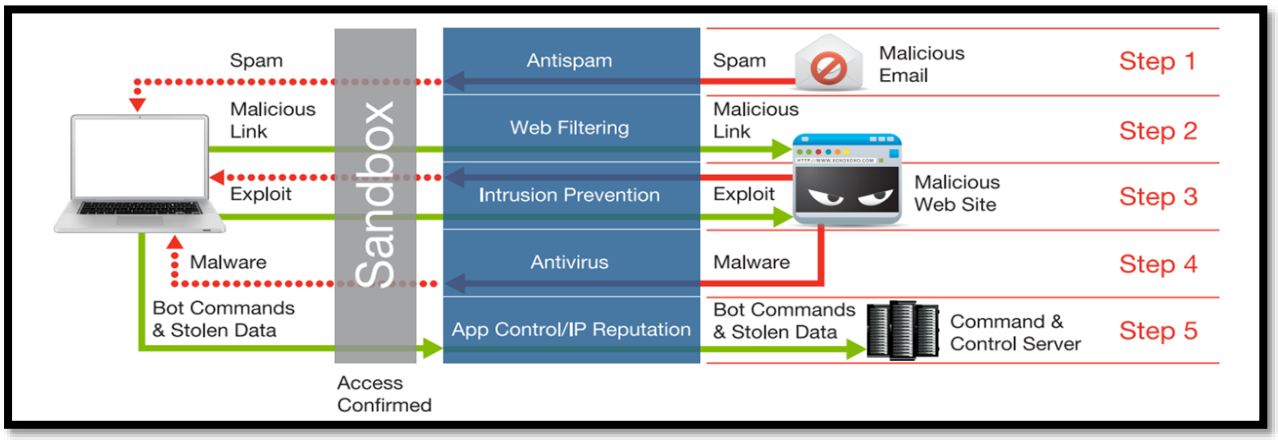


Figure 3. Detailed steps to perform phishing

Step 1: An attacker starts with reconnaissance on the target. They then craft a clever email, often with a malicious link (or file) in it, and send the email to the target. This is where antispam/antiphishing solution may block the email. But if it doesn't: the email goes to the target where the attacker hopes the target will click on the malicious link.

Step 2: If the target clicks on the link, traffic will go out to a web site to establish communication. This is where web filter may block the traffic but if it doesn't: that malicious web site starts attacking the organization.

Step 3: The malicious web site will usually launch exploit attacks at the target to gain access to the system. This is where intrusion prevention system (IPS) attempts to block the attack, but if it doesn't: a tunnel is opened up and the malicious site can launch malware into your organization.

Step 4: With malware seeking entry, ideally your antimalware will protect you, but if it doesn't the attacker gets executable code into target's system where it can run.

Step 5: Once the malicious code is running, it usually looks to access credentials, move laterally in search of sensitive data and collect/stage it within the organization. But in order to complete its mission, it needs to infiltrate that data out to a command & control server. This is where application control, IP reputation, botnet and other protections come into play. But if these technologies don't block this traffic: the target is breached.

Section 1.06 User's Impotence

The following technical inabilities of users leads to Phishing:

- A usable design must consider what humans do well and what they do not do well. Browsers often warn users when they submit form data over an unencrypted connection. This warning is so common that most users ignore it, and some turn the warning off entirely.
- Once a secret has been left unprotected, even for a short time, there is no way to guarantee that it cannot been exploited by an attacker.
- Users can not reliably correctly determine sender identity in email messages.
- Users can not reliably distinguish legitimate email and website content from illegitimate content that has the same "look and feel".
- Users can not reliably parse domain names.
- Users can not reliably distinguish browser chrome from web page content.
- Users can not reliably distinguish actual security indicators from images of those indicators.

- Users do not reliably notice the absence of a security indicator.
- Users can not reliably distinguish multiple windows and their attributes.
- Users do not reliably understand SSL certificates.

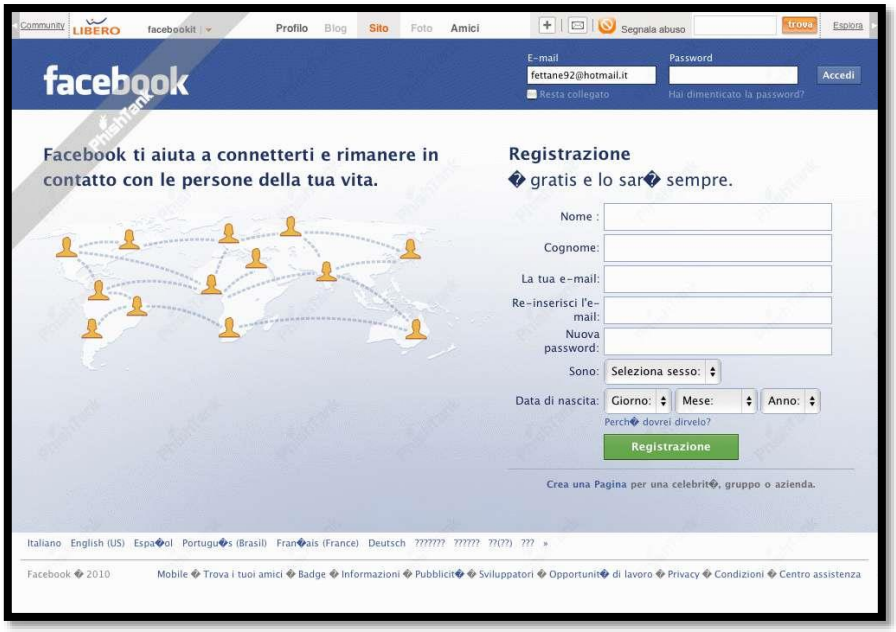


Figure 4. A Phishing website targeting Facebook.

Section 1.07 Phishing Countermeasures

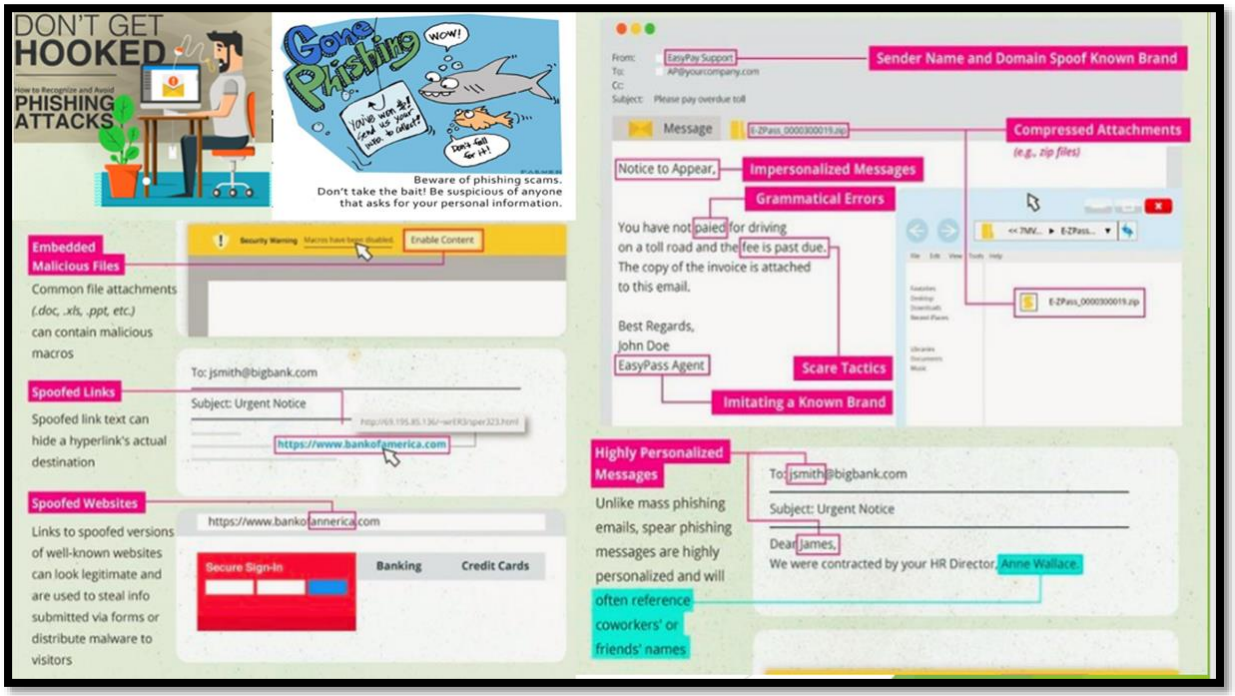


Figure 5. A detailed insight of identifying phishing emails.

Blacklists

Blacklists have been found to be a far more precise mechanism as candidate URLs can undergo much more thorough analysis than can be completed before a page load. Most popular blacklists update every five minutes, so the delay is not significant enough to substantially impact the blacklist's effectiveness.

Browser Plugins and Extensions

A number of clients have been developed to protect users from navigating to fraudulent websites on their computers. In mid 2000s, Anti-Phishing toolbars were in- credibly popular and most made use of a small number of heuristics for detecting Phish in addition to a blacklist of known Phishing sites. Heuristics might include checking whether the page is served over a non-standard port or whether the page has an SSL certificate. Now, extensions are the new heat in the market. Following are popular anti-phishing extensions:

- **Phishing Boat**

  Phishing Boat will scan each link that you click on to make sure that you have visited that page before. If you haven't, then it will ask you to confirm that you want to visit that website. If you confirm, then you will proceed to the website. If you don't confirm, then you will stay on the current page. Because most computer users only visit a handful of distinct websites, and because most phishing attacks rely on a distracted victim visiting a website with an intentionally-misspelled name, Phishing Boat will prevent computer users from making the type of hasty mistakes that enable most phishing attacks.

- **Netcraft**

  The Netcraft Extension is a tool allowing easy lookup of information relating to the sites you visit and providing protection from Phishing. As soon as the first recipients of a phishing mail report it, we can block it for all users of the extension providing an additional level of protection from Phishing. Netcraft processes reports of fraudulent URLs from a diverse variety of sources and proactively searches for new fraudulent sites.

- **Stop Phishing**

Stop Phishing extension will alert you if you open a fraudulent email in your webmail or a fraudulent website in your Chrome browser. You can also report any phishing email received in your webmail to the Stop Phishing users community. Each Phishing report is verified by VERIFROM, when phishing is confirmed each user will be protected against the reported threat.

- **Password Alert**

  If you enter your Google Account password or Google for Work password into anywhere other than Google's sign-in page, you'll receive an alert, so you can quickly change your password if needed. Password Alert also checks each page you visit to see if it's impersonating Google's sign-in page, and alerts you if so.

- **Nohack by Apozy**

  A simple, lightweight and unobtrusive way to make all websites safer. Apozy sandboxes malicious sites to stop phishing attacks, ransomware, and data leaks. Apozy does not interrupt your browsing habits while protecting you.

Article VI. noPhish

noPhish was developed in five stages and they were following:

Stage 1: Develop a prototype of Chrome Extension.

Stage 2: Develop a code comparator, that predicts whether a website is Phishing or not, based solely on the code present in the website's source page.

Stage 3: Automated Feature Extraction.

Stage 4: Develop a XGBOOST binary classifier, to predict whether a website is Phishing or not, based on technical aspects of the website.

Stage 5: Develop a DBSCAN Clustering model, combined with computer vision techniques, to predict whether a website is Phishing or not, based on the visual aspect of the website.

The next articles will give detailed insight of each stage.

Article VII. Stage 1

Section 1.08 Chrome Extensions

Extensions are small software programs that customize the browsing experience. They enable users to tailor Chrome functionality and behavior to individual needs or preferences. They are built only on web technologies such as HTML, JavaScript, and CSS. An extension must fulfill a single purpose that is narrowly defined and easy to understand. A single extension can include multiple components and a range of functionality, as long as everything contributes towards a common purpose.

Extension files are zipped into a single .crx package that the user downloads and installs. This means extensions do not depend on content from the web, unlike ordinary web apps. Extensions are distributed through the Chrome Developer Dashboard and published to the Chrome Web Store.

The project was to build a Google Chrome extension, noPhish, whose one and only functionality was to prevent user from accessing phishing websites. The algorithmic implementation of that functionality was supposed to be programmed in Python. As mentioned above, extension cannot be programmed on Python, so had to be implement over Client-Server architecture.

Section 1.09 Client-Server Architecture

The client–server model is an application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and

electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.

For noPhish, the client is any user who installs it and the server is any system where the Level 1 and 2 models will predict the website's phishing status and return it back to the client.

Section 1.10 Implementation

Extensions can do quite a lot. They use either page actions or browser actions. They can't use both. A page action is a chrome extension that is specific to certain pages. A browser action is not specific to a page and is relevant no matter where one is in the browser.

noPhish extension was built using following steps:

Step 1: Create a manifest file

The manifest.json file tells Chrome important information about the extension, like its name and which permissions it needs. Following is the most basic implementation of manifest file, with all the necessary fields filled in.

```
{
  "manifest_version": 2,
  "name": "noPhish",
  "version": "0.1"
}
```

Step 2: Content scripts

A content script is "a JavaScript file that runs in the context of web pages." This means that a content script can interact with web pages that the browser visits.

noPhish has a content script named content.js which would alert users if the website is phishing or not:

```
// content.js
if Phishing is True:
   alert("POSSIBLE ATTEMPT AT PHISHING!")
```

To inject the script, we need to tell our manifest.json file about it, this to added to manifest.json file:

```
"content_scripts": [
  {
```

```
    "matches": [
      "<all_urls>"
    ],
    "js": ["content.js"]
  }
]
```

This tells Chrome to inject content.js into every page we visit using the special <all_urls> URL pattern. Every single page client visits now pops up an alert if the condition is true.

Step 3: Message Passing

A content script has access to the current page, but is limited in the APIs it can access. For example, it cannot listen for clicks on the browser action. One needs to add a different type of script to our extension, a background script, which has access to every Chrome API but cannot access the current page. As Google puts it:

"Content scripts have some limitations. They cannot use chrome.* APIs, with the exception of extension, i18n, runtime, and storage."

When a new tab is opened, the content script should pull a URL out of the current page and need to hand that URL over to the background script to do something useful with it. In order to communicate, noPhish uses what Google calls message passing, which allows scripts to send and listen for messages. It is the only way for content scripts and background scripts to interact. Adding the following to tell manifest.json about the background script:

```
"background": {
  "scripts": ["background.js"]
}
```

Now, add background.js:

```
// background.js
// Called when the user opens a new tab
chrome.tabs.onUpdated.addListener(function (tabId, changeInfo, tab) {
  // Perform some task on the new tab opened. For example, send a message to the active tab
        chrome.tabs.query({active: true, currentWindow: true}, function(tabs) {
                var activeTab = tabs[0];
        chrome.tabs.sendMessage(activeTab.id, {"message": "new_tab_opened"});
                });
```

```
});
```

This sends an arbitrary "message" that a new tab was opened. Now, need to listen for that message in content.js:

```
// content.js
chrome.runtime.onMessage.addListener(
  function(request, sender, sendResponse) {
    if( request.message === " new_tab_opened " ) {
      var firstHref = $("a[href^='http']").eq(0).attr("href");
      console.log(firstHref);
    }
  }
);
```

Step 4: Using jQuery AJAX to connect to Server

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It makes everything easier. noPhish used the latest minified version, jquery-2.1.3.min.js. To load it, add it to manifest.json before "content.js".

The new manifest.json should look like this:

```
{
  "manifest_version": 2,
  "name": "noPhish",
  "version": "0.1",
  "content_scripts": [
    {
      "matches": [
        "<all_urls>"
      ],
      "js": ["jquery-2.1.3.min.js", "content.js"]
    }
  ]
}
```

To connect to server, the following snippet of code was added to background.js:

```
var urlParam = "https://127.0.0.1:4000/"; //URL where the server is running
$.ajax({
    url: urlParam + "?URL1=" + URL1, //Passing URL1 and URL2 as parameters to Flask
    type: "POST", //Type of Call
    dataType: 'text',
    async: false,
    success: function (response) { //Function when the call was successful. Return value is in
'response'
        console.log(response);
        returnValueFromTextScraper = response;
        return response;
    },
    error: function (error) { //Function when the call was unsuccessful.
        console.log(error);
        returnValueFromTextScraper = "Error"
        return "Error";
    }
});
```

Step 5: Wrapping it up

The full content.js and background.js are above. Here's the full manifest.json:

```
{
  "manifest_version": 2,
  "name": "noPhish",
  "version": "0.1",
  "background": {
    "scripts": ["background.js"]
  },
  "content_scripts": [
    {
      "matches": [
        "<all_urls>"
```

```
    ],
    "js": ["jquery-2.1.3.min.js", "content.js"]
   }
  ],
  "browser_action": {
   "default_icon": "icon.png"
  }
}
```

Step 6: Load Extension into Chrome

To load extension in Chrome, open up chrome://extensions/ in your browser and click "Developer mode" in the top right. Now click "Load unpacked extension…" and select the extension's directory.

noPhish's prototype was successfully built.

- *Figure 6. The extensions pops up a warning to a user when he/she visits a phishing site of www.google.co.in*

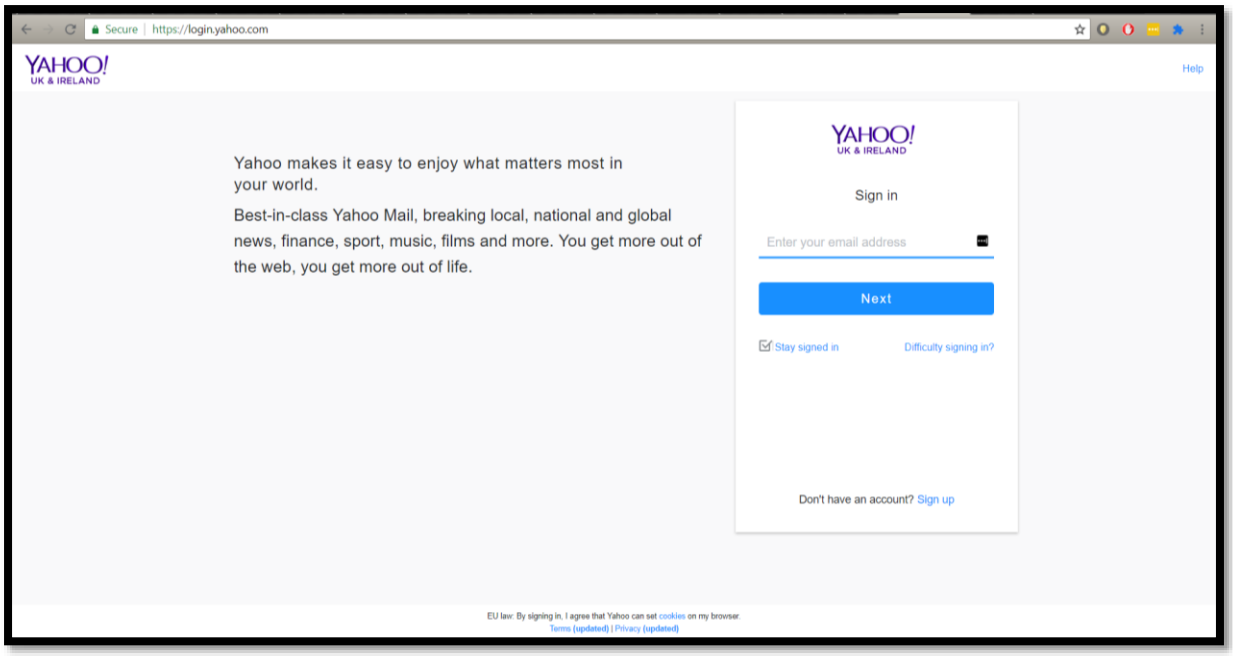


Figure 7. Legitimate page of Yahoo Login. No warning is issued by the extension (blue color extension in address bar) so it correctly identified it as Legitimate website.

Figure 8. Phishing page of Yahoo Login. A warning is issued by the extension (blue color extension in address bar) so it correctly identified it as Phishing website.

And here's the full directory structure:

```
├── background.js
├── content.js
├── icon.png
├── jquery-2.1.3.min.js
└── manifest.json
```

Article VIII. Stage 2

In Stage 2, a code comparator was built to find how similar the underlying codes are of a legitimate site and its corresponding phishing site.

Section 1.11 Data Selection

The phishing and legitimate URLs that was supposed to be selected for feature extraction was collected from following resources:

Phishing URLs: Phishtank

It is an online repository containing a massive collection of known phishing websites. It has information about the phishing sites such as URL, targeted brand (which brand it targeted), online (is it still active) and verified (verified by an authority) regarding the website.

Legitimate URLs: Alexa

Founded in 1996, Alexa is a global pioneer in the world of analytical insight. It is the most robust and accurate web analytics service of any provider. Their global traffic rank is a measure of how a website is doing relative to all other sites on the web over the past 3 months. The rank is calculated using a proprietary methodology that combines a site's estimated average of daily unique visitors and its estimated number of page-views over the past 3 months.

A script was written to get the URLs from Phishtank and Alexa and dump it into a CSV file. URLs from Phishtank were only added to the list if they were bot “Online” and “Verified”.

Section 1.12 Development

The algorithm was written in Python using Jupyter Notebook as an interactive editor using following libraries:

```
from flask import Flask, request, json
from bs4 import BeautifulSoup
import difflib
import urllib.request
```

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

Extract the URL's from for an input website

```
from bs4 import BeautifulSoup
import requests
url = raw_input("Enter a website to extract the URL's from: ")
r  = requests.get("http://" +url)
data = r.text
soup = BeautifulSoup(data)
for link in soup.find_all('a'):
print(link.get('href'))
```

Extract HTML content of URL specified and compare the title and domain of the website

```
import urllib.request
from bs4 import BeautifulSoup

first_url = "https://www.netflix.com/in/"
domainName = first_url.split("//")[-1].split("/")[0].split('?')[0].split(".")[0] #Extract domain name
page1 = urllib.request.urlopen(first_url) #Get URL HTML contents
text1 = BeautifulSoup(page1, 'html.parser') #Convert to BeutifulSoup
textx = str(text1.prettify()) #Convert the HTML in its form
titleOfPage = text1.title.string) #Extract title of the page from HTML <title> tag in <head>
if(titleOfPage == domainName):
print("Same Title as the Domain")
else:
print("Different Title as the Domain")
```

Similarly, to extract iframes

```
for iframe in iframexx:
response = urllib2.urlopen(iframe.attrs['src'])
iframe_soup = BeautifulSoup(response)
```

The difflib library contains tools for computing and working with differences between sequences. It is especially useful for comparing text, and includes functions that produce reports using several common difference formats.

A unified diff includes only the modified lines and a bit of context. The unified_diff() function produces this sort of output.

```
//difflib_unified.py
import difflib
from difflib_data import *

diff = difflib.unified_diff(
text1_lines,
text2_lines,
lineterm='',
)
print('\n'.join(diff))
```

Section 1.13 Testing

The algorithm was tested rigorously, both programmatically and manually. As the web (page source) scraping depends upon the underlying code written for the legitimate and phishing website, the correlation between them was hard to figure out because of the following problems:

- same website can be written in many different combination of scripting languages,
- rearrangement of the modules doesn't change the website functionalities, and
- naming problem, that is the same Image can be saved with a different name.

Hypothetically, it was assumed that websites which looked similar would have at least some co-relation between their underlying coding patterns despite the above problems, but the hypothesis failed. There were very rare cases where the legitimate website's code was similar

to corresponding website's code. The following are two of the many test cases:

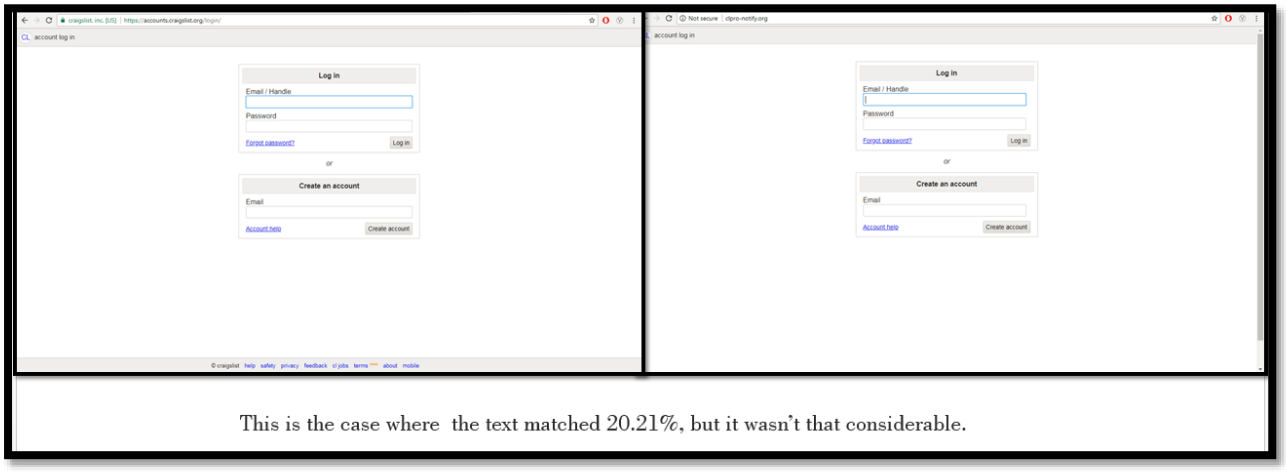


Figure 9. 20.21% was the highest similarity achieved. This was not significant because many legitimate to legitimate website comparison also lay around that figure. :

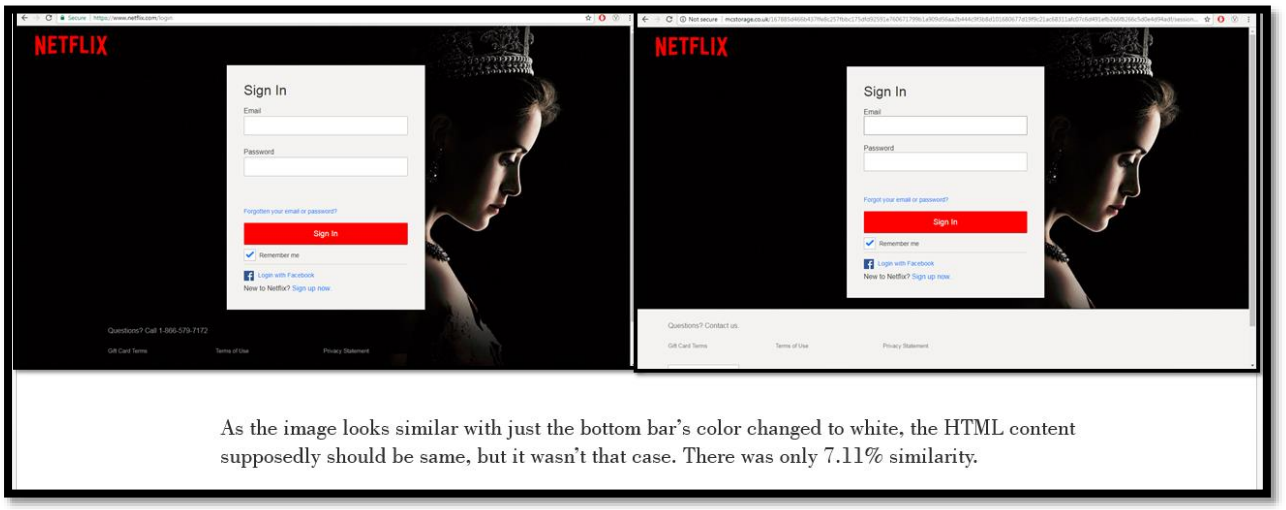


Figure 10. The webpages looks very similar but the underlying code wasn't so it giva such a low similarity.

Thus, it was deduced that the assumptions that were made had some flaws and thus, couldn't provide the expected results. So code comparator was not a useful technique to find if a website is phishing or not. Therefore, this technique was not incorporated in noPhish.

Article IX. Stage 3

At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. If one has many independent features that each correlate well with the class, learning is easy. On the other hand, if the class is a very complex function of the features, one may not be able to learn it. Often, the raw data is not in a form that is amenable to learning, but one can construct features from it that are. This is typically where most of the effort in a machine learning project goes. It is often also one of the most interesting parts, where intuition, creativity and "black art" are as important as the technical stuff.

It is surprising that little time in a machine learning project is spent actually doing machine learning, but it makes sense if one considers how time-consuming it is to gather data, integrate it, clean it and pre-process it, and how much trial and error can go into feature design. Also, machine learning is not a one-shot process of building a dataset and running a learner, but rather an iterative process of running the learner, analyzing the results, modifying the data and/or the learner, and repeating. Learning is often the quickest part of this, but that's because we've already mastered it pretty well! Feature extraction is more difficult because it's domain-specific, while learners can be largely general-purpose.

noPhish has two level validation mechanism which predicts based on various features of the website. The features to be extracted were novel and had not been used by any previous anti-phishing application before (In few cases, only small subsets of the features were used by some researchers but that didn't fulfill the complete purpose of noPhish). Thus, a completely new script was written to extract various features of a website, with URL as the only input. This feature extraction was supposed to done on huge number of URLs, so an automated script was written. The automated feature extraction script should be completely automated and have a capacity to run for days, without intervention, along with solving all the errors and exceptions that occur on the fly.

Section 1.14 Data selection

The CSV created for Stage 2 was reused here.

Section 1.15 Features

The features chosen for Level 1 was on the technical aspects of the website. "Technical aspect" refers to the underlying code and other technical details.

The following table describes important technical features that were extracted from a website for Level 1 model.

Source	Features
Initial URL	Domain tokens, path tokens, query parameters, is obfuscated?, number of subdomains, length of domain, length of path, length of URL (From here on out, we denote this list as URL features)
Redirects	URL features for each redirect, number of redirects, type of redirect
Frame URLs	URL features for each embedded IFrame
Source URLs	URL features for every outgoing network request; includes scripts, redirects, and embedded content
HTML Content	Tokens of main HTML, frame HTML, Spelling match, Copying Website, Disabling Right Click, Submission of info using submit button and script content
Page Links	URL features for each link, number of links
JavaScript Events	Number of user prompts, tokens of prompts, onbeforeunload event present?
Pop-up Windows	URL features for each window URL, number of windows, behavior that caused new window
Plugins	URL features for each plugin URL, number of plugins, application type of plugin
HTTP Headers	Tokens of all field names and values; time-based fields are ignored
Encryption and Security	Using SSL Certificate, CCIA, Anomalous cookie detail, Unique name certificate
DNS	Reverse IP to host match?
Geolocation	Country code, City code (if available) for each IP encountered

TABLE 1: List of features

The features chosen for Level 2 was on the visual aspect of the website. “Visual aspect” refers to screenshot taken of the website. The feature was extracted using image processing

techniques such as grayscale convertor, image resizing and image hashing algorithm. These techniques were implemented on the screenshot taken of the webpage. Thus, the features were derived from the knowledge gained out of the pixel values of the screenshot image. The data to be extracted from the image was the perceptual hash (pHash) of the image.

pHash was calculated not only on the complete image (singleHash), but also on the sections of the image(multipleHash), so that the image's overall integrity is captured along with its minor details. The perceptual hash calculated for the complete image is known as "singleHash", while the perceptual hash calculated for the sections of the image is known as "multipleHash". Image was divided into four equal sections and perceptual hash was calculated for each one of them. The perceptual hash is calculated using following steps:

- Resize the image (or section) in 16*16 dimensions which will have 256 pixels worth information. Increasing the dimensions will lead to greater hash length.
- Find the mean pixel value of the image (or section) by summing up all the 256 pixels and divide it by 256.
- Using the mean pixel value, represent each pixel value by binary digit. If the pixel value is more than mean pixel value, then the binary representation is 1, or else 0.
- Thus, this will create a hash of 256 length.

Thus, each image had 5 pHash features – singleHash (pHash of complete image) and 4 differentialHash (pHash of 4 equally divided sectors).

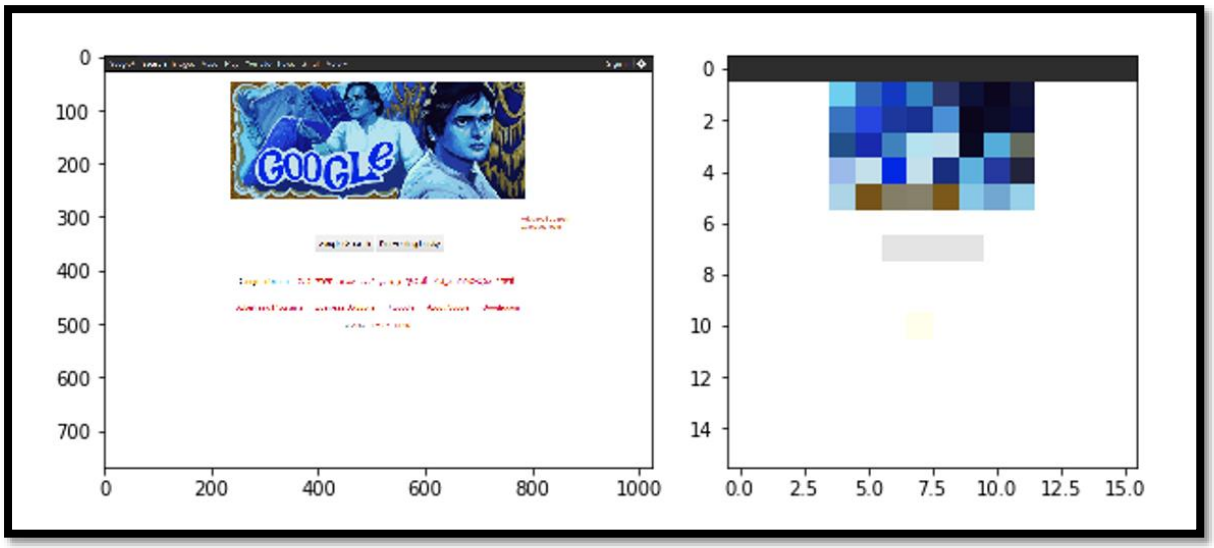


*Figure 11. Left Image: Original image of www.google.com (Google) having a dimension of 1024*768. Right Image: Plot of resized image having dimension of 16*16 for singleHash.*

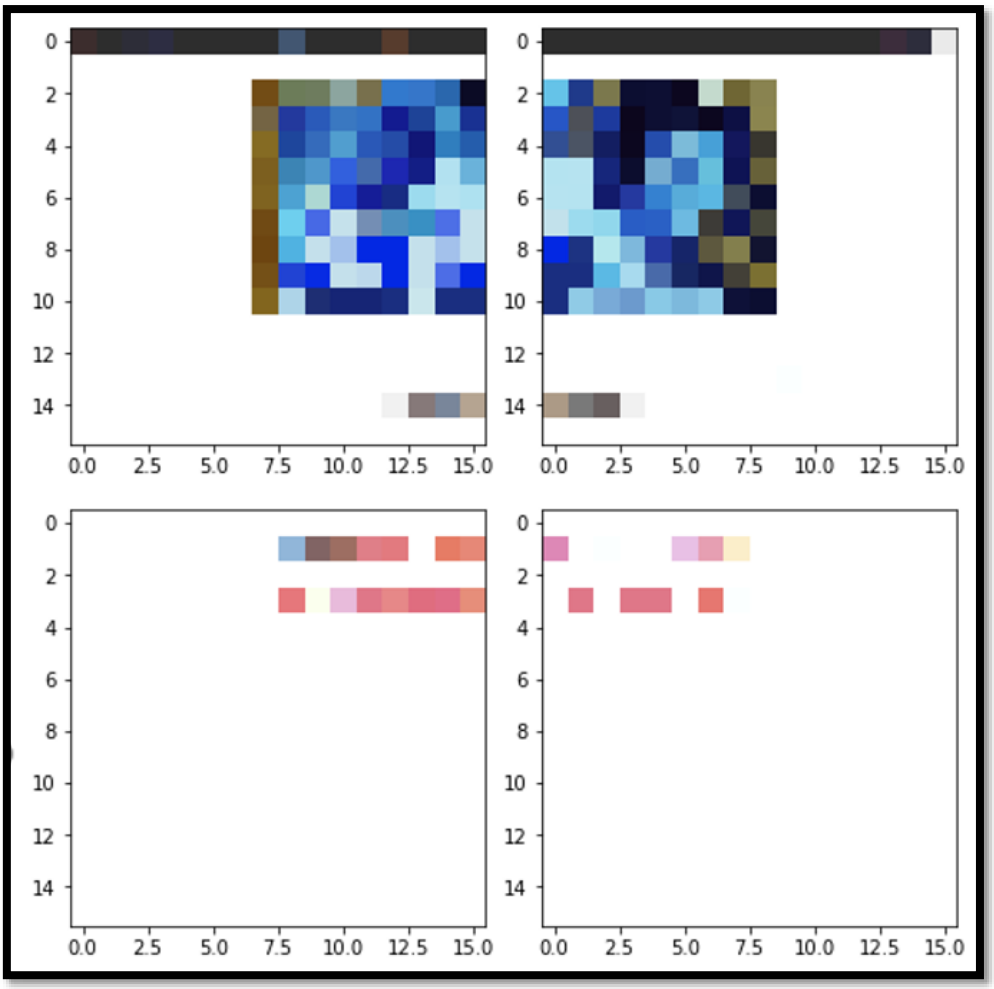


*Figure 12. Visualization of Original Image when divided into 4 sections and each resized in dimension of 16*16.*

Section 1.16 Preprocessing Data

Machine learning algorithms learn from data. It is critical that one feeds them the right data for the problem one wants to solve. Even if one has good data, one needs to make sure that it is in a useful scale, format and even that meaningful features are included.

- **Formatting:** The data collected from feature extraction script was saved in Comma Separated Format (CSV) file. The machine learning model that I was going to implement had a constraint that it only takes Pandas DataFrame or Numpy array as an input. Thus, the data was not in a format that was suitable for a model to work with and had to be transformed into that format.
- **Cleaning:** Cleaning data is the removal or fixing of missing data. There were occasions where the data was incomplete or did not carry any value. These instances had to be resolved, and were taken care by filling up the missing entries with either 0 or 1 or infinite value (such as -9999) as per the feature's description and range values.

- **Binarize Data:** One can transform data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0. It was useful when I had to make mark certain similarities and dissimilarities.
- **Text Mining:** The machine learning models are generally not designed to accept string data values, so the features which has string as their data type have to be transformed into integer data type. When the string value is converted to integer value, one has to make sure that the information that string provided is replicated in integer values too, and does not get lost. This was done using following two techniques:
    - **TFIDF:** In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. For noPhish, each website is a document and set of all websites is a corpus.
    - **One Hot Encoding:** One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. The categorical value represents the numerical value of the entry in the dataset. As the number of unique entries increases, the categorical values also proportionally increase. The categorical values start from 0 goes all the way up to N-1 categories.

Section 1.17 Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. Feature Engineering is an art.

Feature engineering is the most important art in machine learning which creates the huge difference between a good model and a bad model. Let's see what feature engineering covers. Suppose, we are given a data "domainCreationDate vs domainExpirationDate". Then, we have to predict whether the website is phishing or not.

domainCreationDate	domainExpirationDate	PhishingOrNot
2017-02-20	2017-05-20	Phishing
2015-04-23	2018-04-23	Not Phishing
2013-02-11	2016-09-11	Not Phishing

As the status of the flight directly depends on the "domainLife" (domainCreationDate - domainExpirationDate), and not directly on the "domainCreationDate" and "domainExpirationDate", a new feature "domainLife" will be created and this is called Feature Engineering. Feature engineering turn your inputs into things the algorithm can understand. Using the "domainLife" feature, the machine will learn better as this feature is directly related to the PhishingOrNot of the website.

Section 1.18 Implementation

The automated feature extraction script was written for noPhish.

Input: List of URLs with each having a label of "Phishing" or "Legitimate" (N*2 vector, where is N is the total number of URLs)

Output: CSV file which will have various features – both for Level 1 and 2 – for each input URL.

Section 1.19 Problems Faced While Implementation

While automating a feature extraction script, there were few challenged faced and those are as follows:

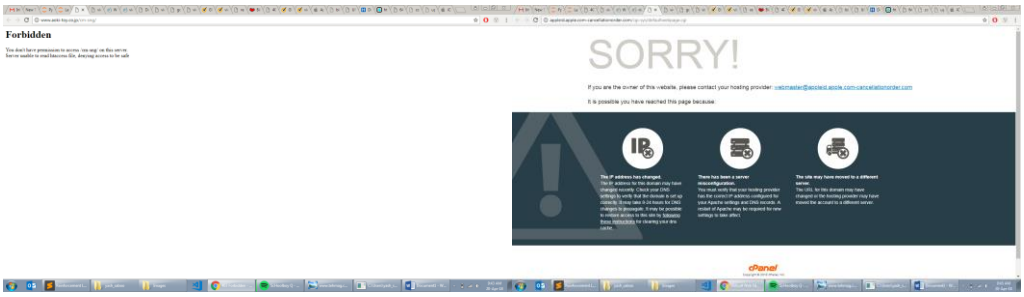
Infinite Time Problem: Sometimes a feature would be present for a website, but it would take infinite time to extract it. This leads to improper stoppage which doesn't even raise an error explicitly. So, a decorator was created in Python called "timeout(numberOfseconds)" which would raise "TimeoutError" using SIGNAL function present in UNIX systems. (Thus, this script only works on UNIX – based systems)

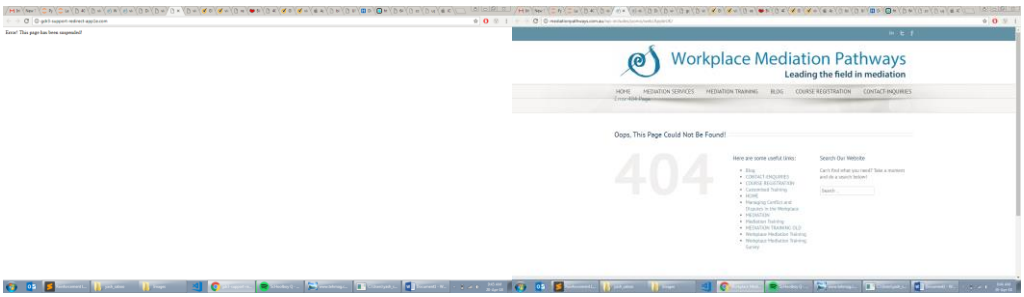
Data Corruption: Some websites produce following errors which corrupts the data:

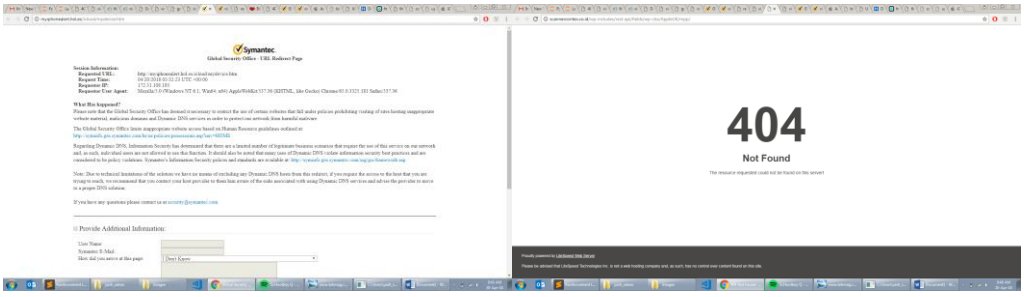
- Throws 403 Error: Forbidden
- Throws 404 Error: Website Not Found
- Redirects to some other site which defeats the purpose of getting data of the original site altogether
- As phishing sites are short-lived so the content for which it was categorized as Phishing has been replaced by some genuine content.
- Account Suspended of the website Domain

- When the URL link received is not available anymore, usually happens with shortened URL.
- When the website is “Sleeping”, usually happens with the websites hosted on some hosting service websites.
- The website blocked by Symantec Network

A separate script was written to find if the website would throw above errors, and if it did then that URL was filtered out from the URL list. Few screenshots of such websites are as follows:







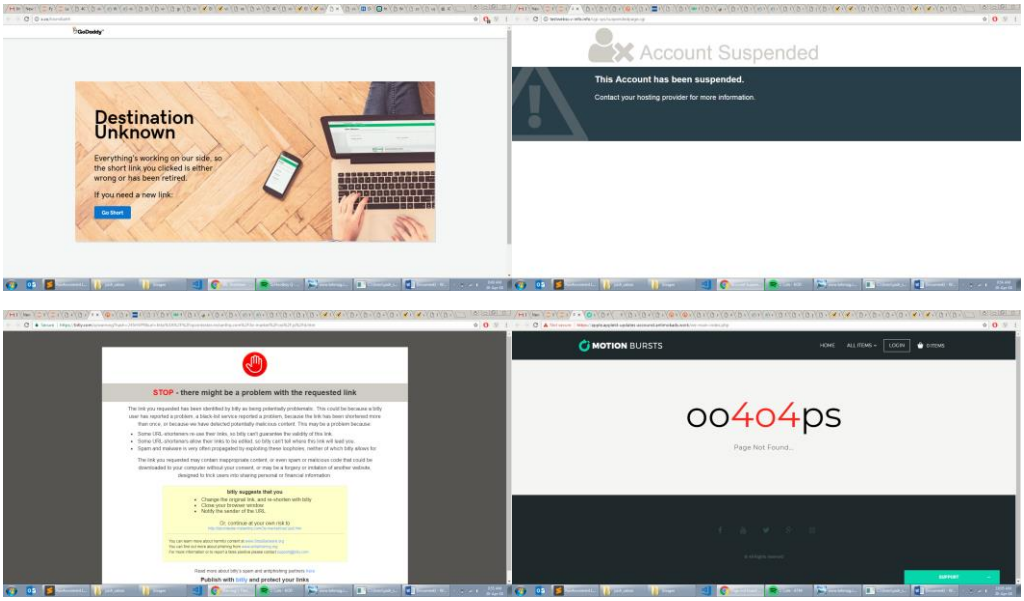


Figure 13. Set of images which leads to Data corruption

Article X. Stage 4

The aim of the Stage 4 was to build a machine learning model that could predict if the website is Phishing or not, based on technical features of the website.

Following concepts were used:

- *Models:* Decisions Trees, Random Forests, XGBoost
- *Learning:* Supervised
- *Data:* Unbalanced
- *Metrics:* Accuracy, Receiver Operating Characteristic(ROC) Curve, Feature Importance Map

XGBoost model was used for binary classification of a website. To understand XGBoost, one needs to understand working of Decision Trees and Random Forests.

Section 1.20 Decision Trees

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, the population or sample is split into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables. The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from training data. Decision Trees are rarely applied because of their over fitting nature which is solved by Random Forest.

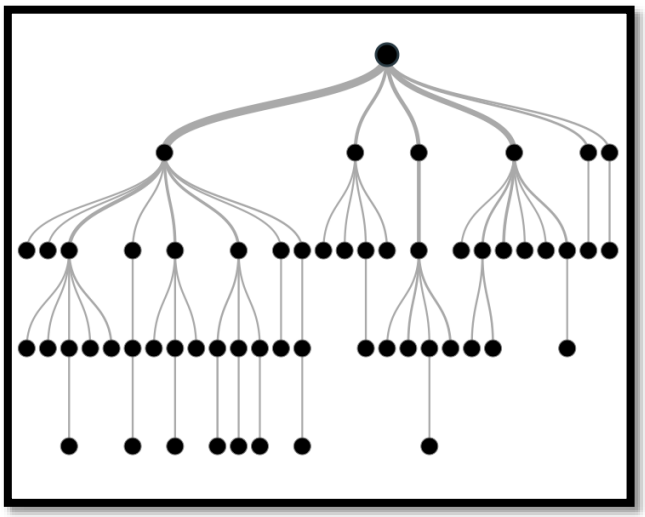


Figure 14. Decision Tree

Section 1.21 Random Forest

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a set of decision trees. In general, the more trees in the forest, the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

Random Forest is considered to be a panacea of all data science problems. Random Forest is a versatile machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and does a fairly good job. Random forest has the power of handling large data set with higher dimensionality. It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.

Figure 15. Random Forest

Section 1.22 Boosting

The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. Let's understand this definition in detail with respect to noPhish:

How would one classify a website as Phishing or not? For a website X the following inference were drawn:

- Website X's URL length is really long. [Phishing]
- Website X's page has many grammatical errors. [Phishing]

- Website X's domain life is less than a year. [Phishing]
- Website X uses a secure connection that is HTTPS. [Legitimate]
- Website X's page doesn't pop-up boxes or prompts to interact with user forcefully. [Legitimate]

Above, these are multiple rules to classify a website into 'phishing or 'legitimate'. But, none of these rules individually are strong enough to successfully classify a website. Therefore, these rules are called as weak learner. To convert weak learner to strong learner, we'll combine the prediction of each weak learner using methods like:

- Using average/ weighted average
- Considering prediction has higher vote

For example, as previously designed, the 5 weak learners can be converted to strong learner. Out of these 5, 3 are voted as 'Phishing' and 2 are voted as 'Phishing'. In this case, by default, we'll consider website X as Phishing because we have higher(3) vote for 'Phishing'.

Gradient Boosting (GBM) and XGBoost are the most efficient and popular boosting techniques. noPhish used XGBoost as the model over GBM because of the following advantages of XGBoost over GBM:

- **Regularization**: Standard GBM implementation has no regularization like XGBoost, therefore it also helps to reduce over fitting. In fact, XGBoost is also known as 'regularized boosting' technique.
- **Parallel Processing**: XGBoost implements parallel processing and is blazingly faster as compared to GBM.
- **High Flexibility**: XGBoost allow users to define custom optimization objectives and evaluation criteria. This adds a whole new dimension to the model and there is no limit to what one can do.

Section 1.23 XGBoost (eXtreme Gradient Boosting)

The XGBoost implements the gradient boosting decision tree algorithm. This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

XGBoost is designed for speed and performance. This approach supports both regression and classification predictive modeling problems.

Section 1.24 Training, Testing and Benchmarking the Model

XGBoost model was finally implemented. Following are the statistics of the model's training and testing:

- Total Websites: 5,125
- Test Accuracy: 92.52%
- Split Ratio: 0.7
- Training Websites: 3587
- Testing Websites: 1538

As the data was unbalanced, accuracy was not the appropriate metric to measure the model's performance. Thus, ROC curve was used as the metric for model performance measure.

Section 1.25 ROC Curve

In statistics, a receiver operating characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the fall-out or probability of false alarm and can be calculated as (1 − specificity). The area under the ROC curve (AUC) of a test can be used as a criterion to measure the test's performance ability. Larger the AUC, better is the performance of the model.

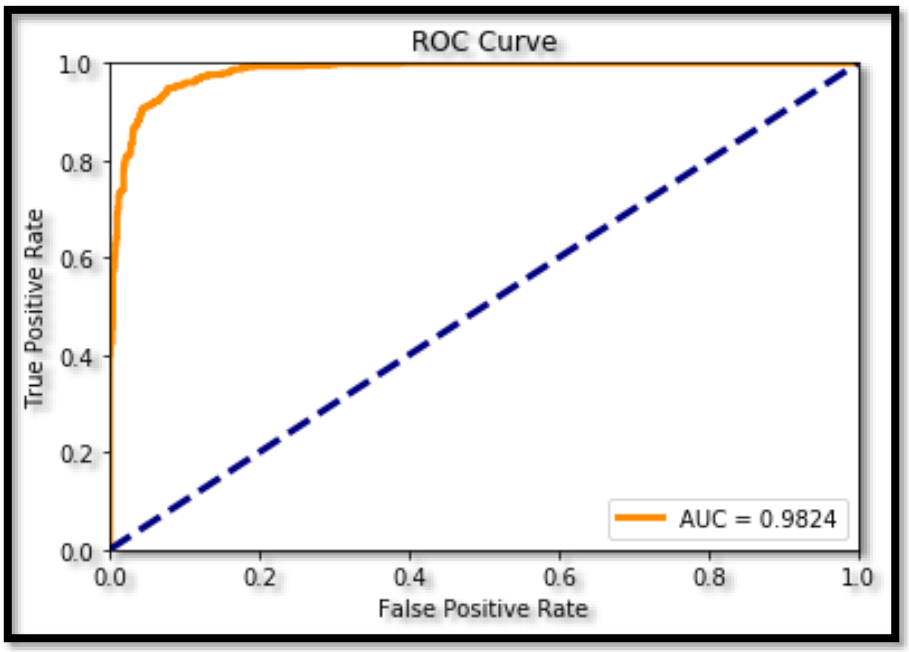


Figure 16. ROC Curve of Level 1 XGBoost model.

As evident from the figure, the AUC for Level 1 is 0.98, which is excellent for a machine learning model.

Section 1.26 Feature Importance Map

XGBoost provides a functionality called "Feature Importance Map" which helps identify important features. It, indirectly, helps find features which were not contributing in the classification process.

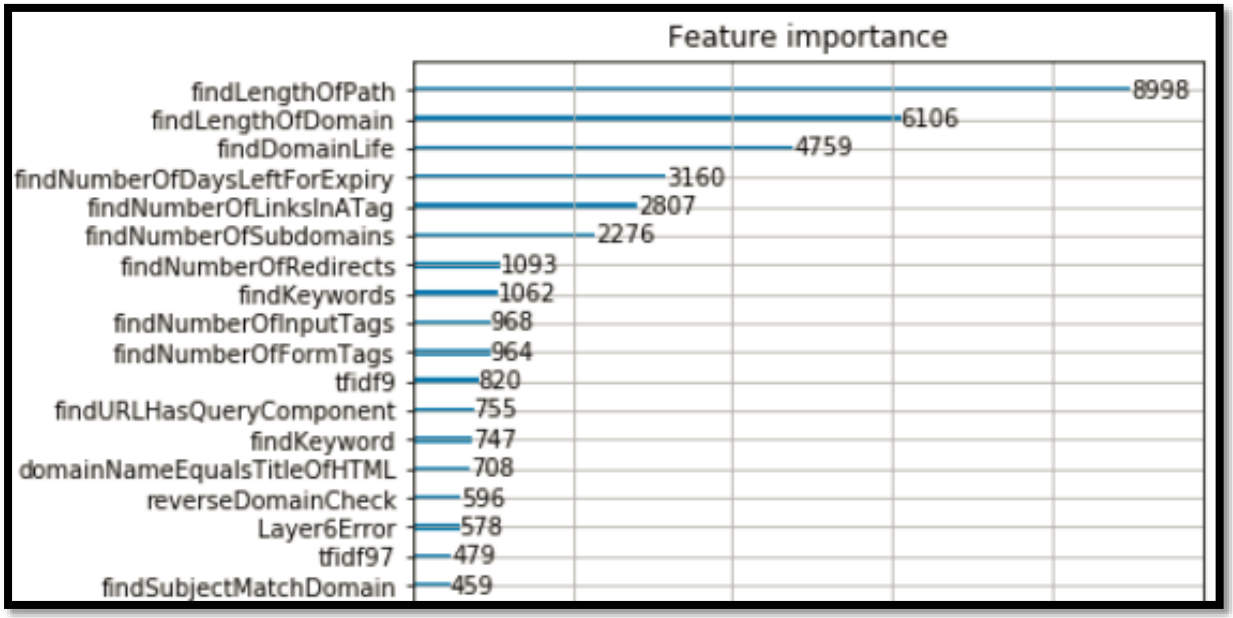


Figure 17. Partial "Feature Importance" chart of Level 1 XGBoost Model.

The score present for each feature represents its importance. Higher the value, higher the importance.

Article XI. Stage 5

The aim of the Stage 5 was to build a machine learning model that could predict if the website is Phishing or not, based on *visual features* of the website.

Following concepts were used:

- *Models:* DBSCAN (Clustering technique)
- *Learning:* Semi-supervised
- *Data:* Unbalanced
- *Metrics:* Confusion Matrix

Level 2 of noPhish was implemented as semi-supervised learning. In semi-supervised learning, labeled data is used to help identify that there are specific groups of webpage types present in the data and what they might be. The clustering algorithm is then trained on unlabeled data to define the boundaries of those webpage types.

Section 1.27 Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters. There is no objectively "correct" clustering algorithm, but as it was noted, "clustering is in the eye of the beholder". The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another.

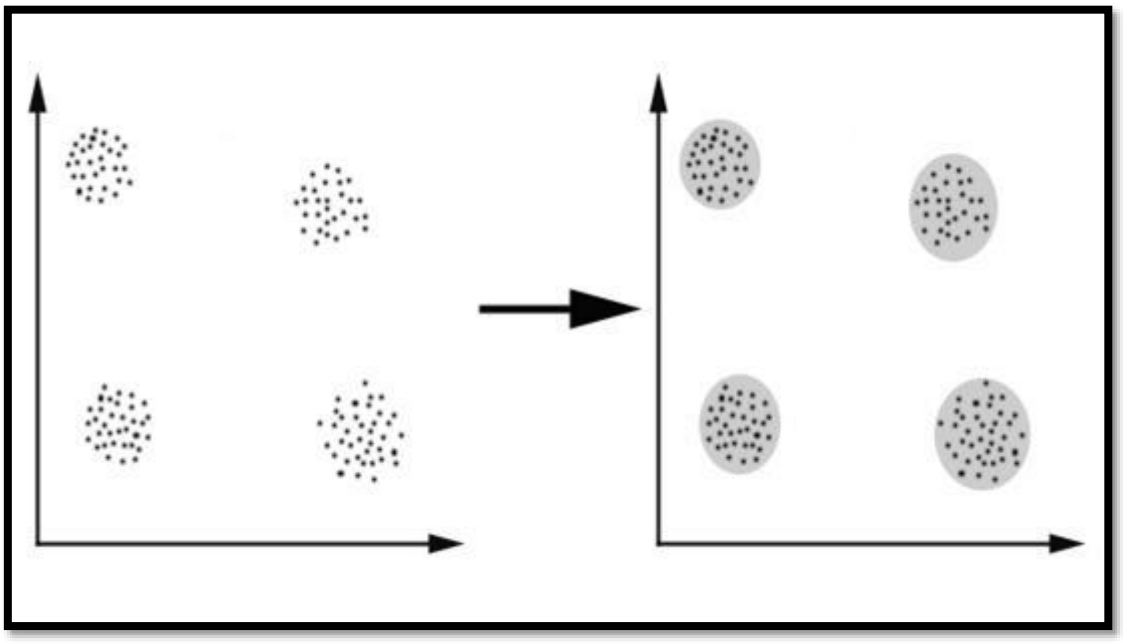


Figure 18. Showing four clusters formed from the set of unlabeled data

This technique was used to cluster similar images so that phishing of one domain can fall into one cluster. To cluster the data, density-based clustering was used.

Section 1.28 Density-based Clustering

These models search the data space for areas of varied density of data points in the data space. It isolates various density regions and assign the data points within these regions in the same cluster. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points. It is based on connecting points within certain distance thresholds. However, it only connects points that satisfy a density criterion, in the original variant defined as a minimum number of other objects within this radius.

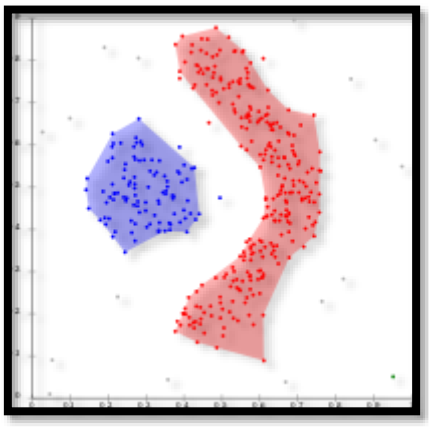


Figure 19. Density-based clustering with DBSCAN. DBSCAN can find non-linearly separable clusters. This dataset cannot be adequately clustered with k-means.

We used a very popular and efficient density-based clustering algorithm called density-based spatial clustering of applications with noise (DBSCAN).

Section 1.29 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

To understand DBSCAN better, consider a set of points in some space to be clustered. For DBSCAN clustering, the points are classified as core points, (density-) reachable points and outliers, as follows:

- A point p is a core point if at least minPts points are within distance ε (ε is the maximum radius of the neighborhood from p) of it (including p). Those points are said to be directly reachable from p.
- A point q is directly reachable from p if point q is within distance ε from point p and p must be a core point.
- A point q is reachable from p if there is a path p1... pn with p1 = p and pn = q, where each pi+1 is directly reachable from pi (all the points on the path must be core points, with the possible exception of q).
- All points not reachable from any other point are outliers.

Now if p is a core point, then it forms a cluster together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.

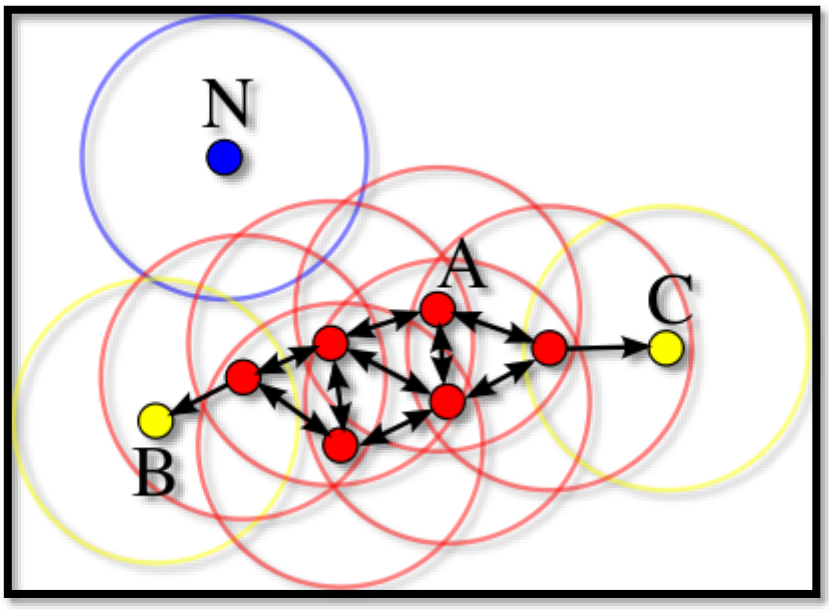


Figure 20. In this diagram, minPts = 4. Point A and the other red points are core points, because the area surrounding these points in a ε radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable. DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means. DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced. DBSCAN has a notion of noise, and is robust to outliers. DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database. The parameters minPts and ε can be set by a domain expert, if the data is well understood.

Section 1.30 Sensitivity and Specificity

To benchmark the algorithms, I found the sensitivity (also called the true positive rate) and specificity (also called the true negative rate) of each algorithm.

Following definitions were used [Labels: Phishing – 1 and Legitimate – 0]:

	Phishing is present	*Phishing is absent*
Detects Phishing	True Positive (TP)	False Positive (FP)
Doesn't Detect Phishing	False Negative (FN)	True Negative (TN)

Sensitivity = TP / (TP + FN)

Specificity = TN / (FP + TN)

Section 1.31 t-SNE

T-distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton. t-SNE is a prize-winning technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. This technique was solely used for visualizing the clusters formed by DBSCAN for Level 2 on training images.

Section 1.32 Training and Testing of the Model

Following are the statistics of the training and testing of the DBSCAN model:

Training:

- Number of Images: 1358
- Number of Training Images: 909
- Number of Testing Images: 449

Testing:

- True Positive: 364
- True Negative: 61
- False Positive: 0
- False Negative: 85
- True Positive Rate: 81.07%
- False Positive Rate: 0.00 %

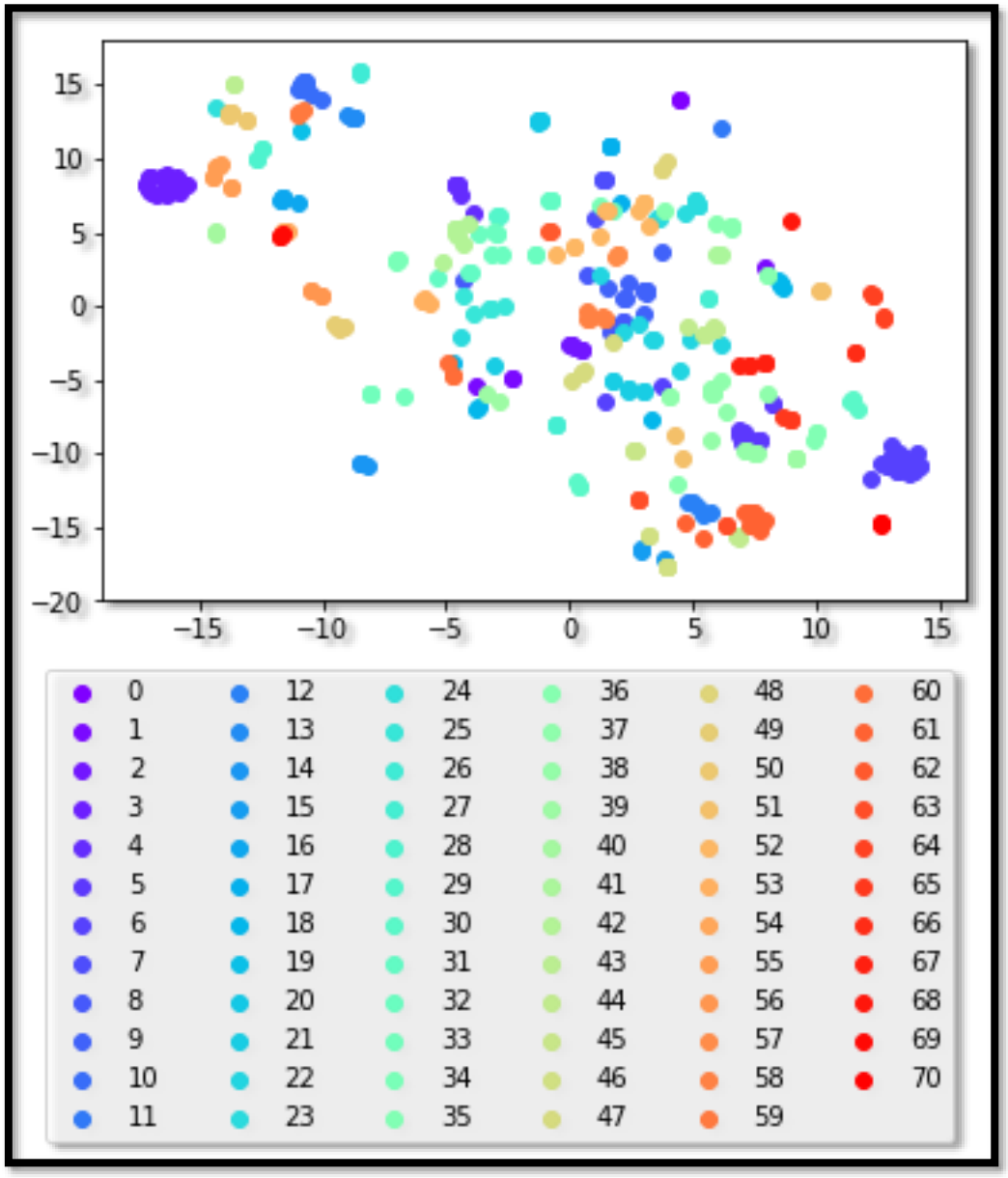


Figure 21. t-SNE dimensionality reduced visualization of the clusters formed by DBSCAN on the training images. The number represented in label denotes the cluster number and each cluster is represented with a color in rainbow scale.

Article XII. Software Tools

The following software tools were used while developing noPhish:

Operating System	Windows 7, Ubuntu Linux 16.0 (For Stage 3)
Web Server	Flask, Wamp
Web Technologies	JavaScript, HTML, CSS
Programming Language	PHP, Python
Web Service	REST
Text Editor	Sublime Text 2, Visual Studio Code, Jupyter Notebook
Version Control	Git
Collaboration	Atlassian Stash

Article XIII. Hardware Tools

There is no specific hardware required but the following hardware tools were used:

Processor	Intel Core i7-6700HQ
Frequency	2.60 GHz
Cores	4
RAM	8 GB
Class	DDR3
Storage	2 TB

Article XIV. Process Model

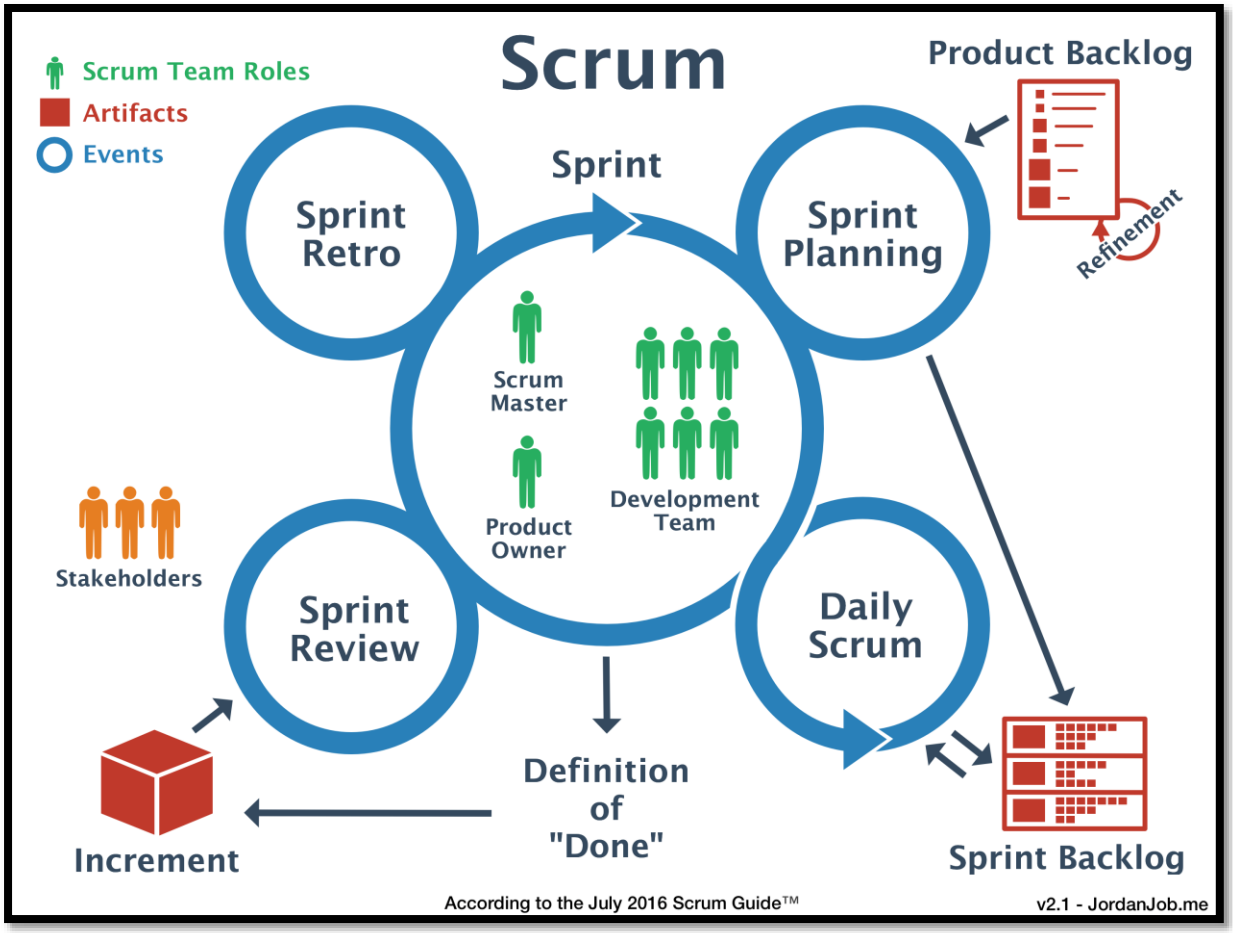


Figure 22. Scrum

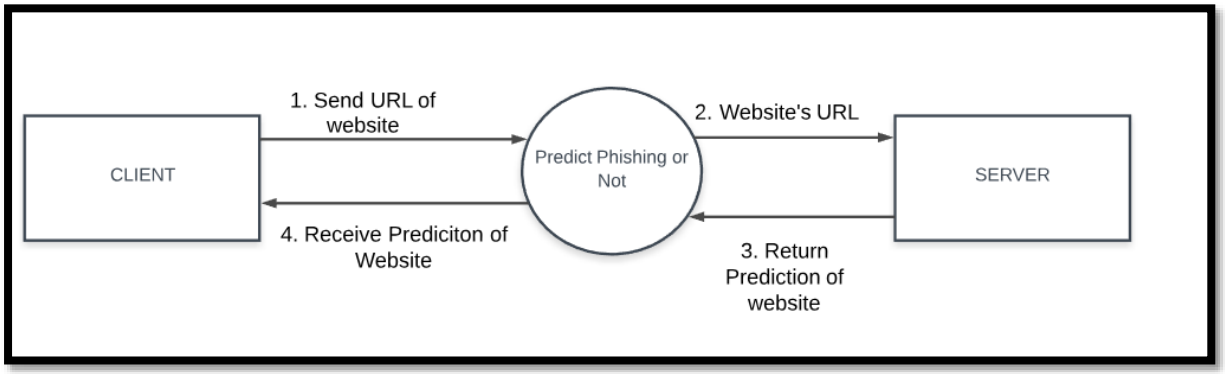
The process model that was used was Scrum. Scrum is an agile framework for managing work with an emphasis on software development. It is designed for teams of three to nine developers who break their work into actions that can be completed within time-boxed iterations, called sprints (two weeks, in my case) and track progress and re-plan in 15-minute stand-up meetings, called daily scrums.

Scrum is an iterative and incremental framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.

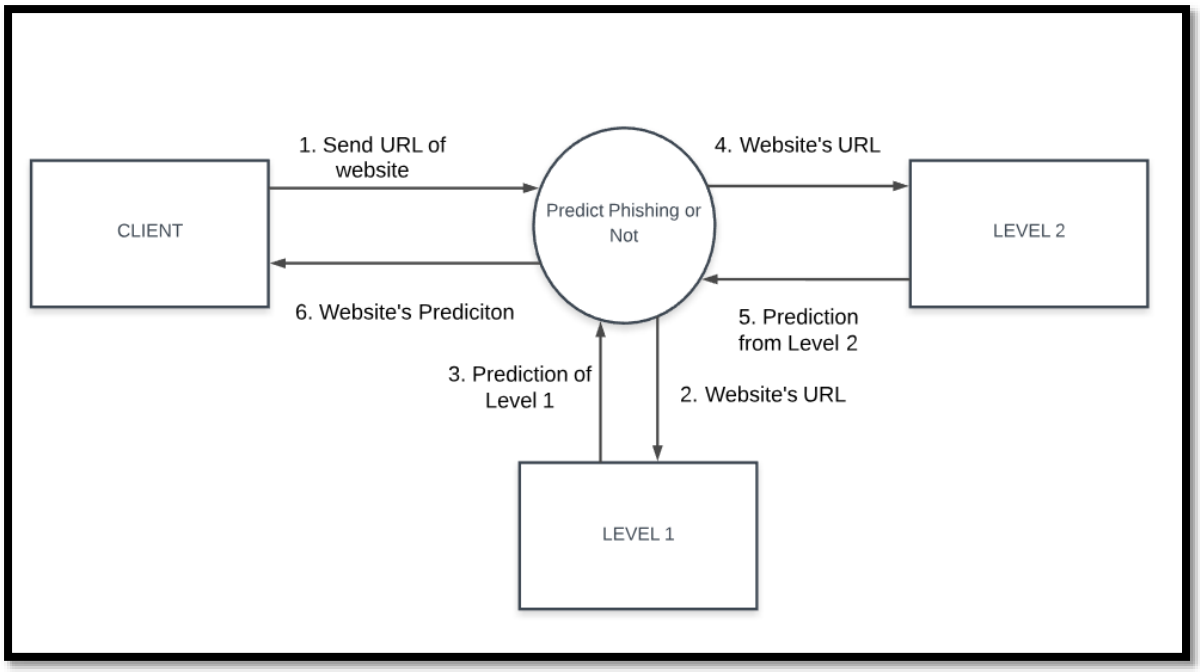
Article XV. Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

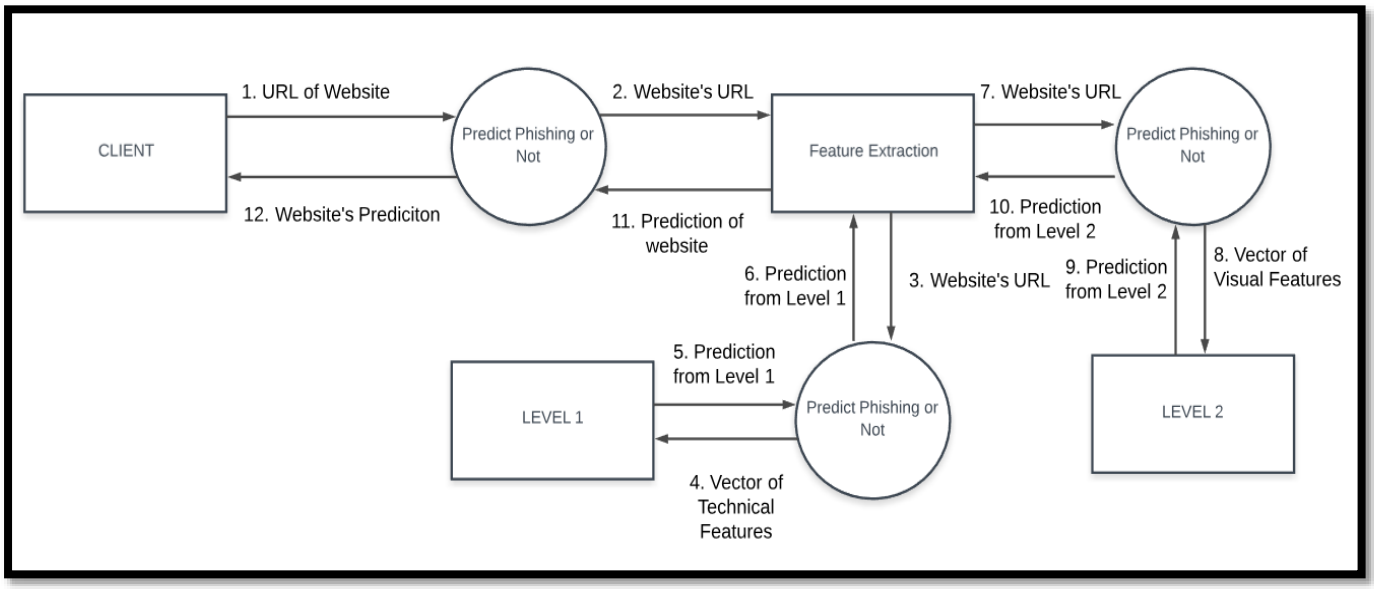
Section 1.33 DFD Level 0



Section 1.34 DFD Level 1

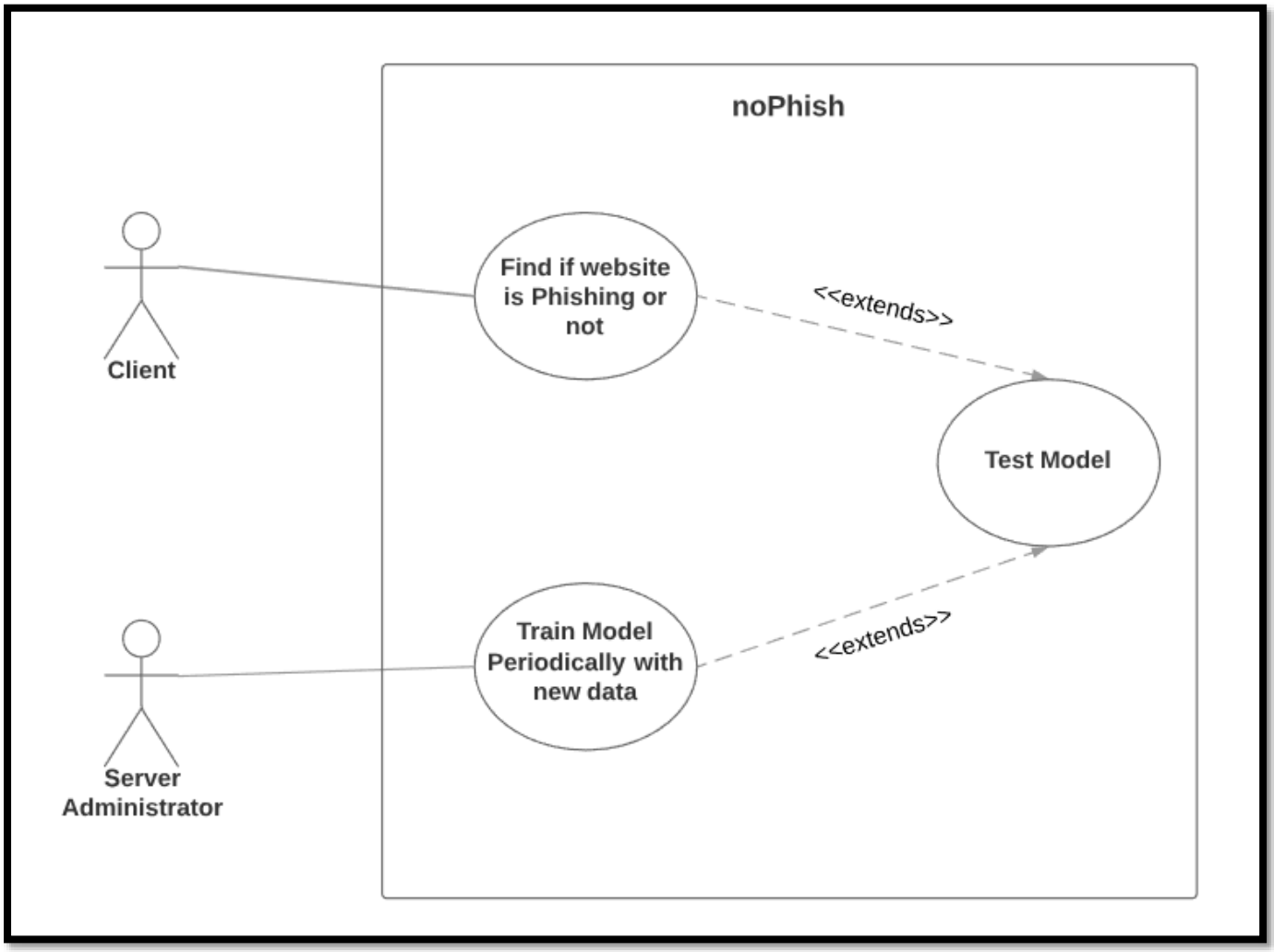


Section 1.35 DFD Level 2



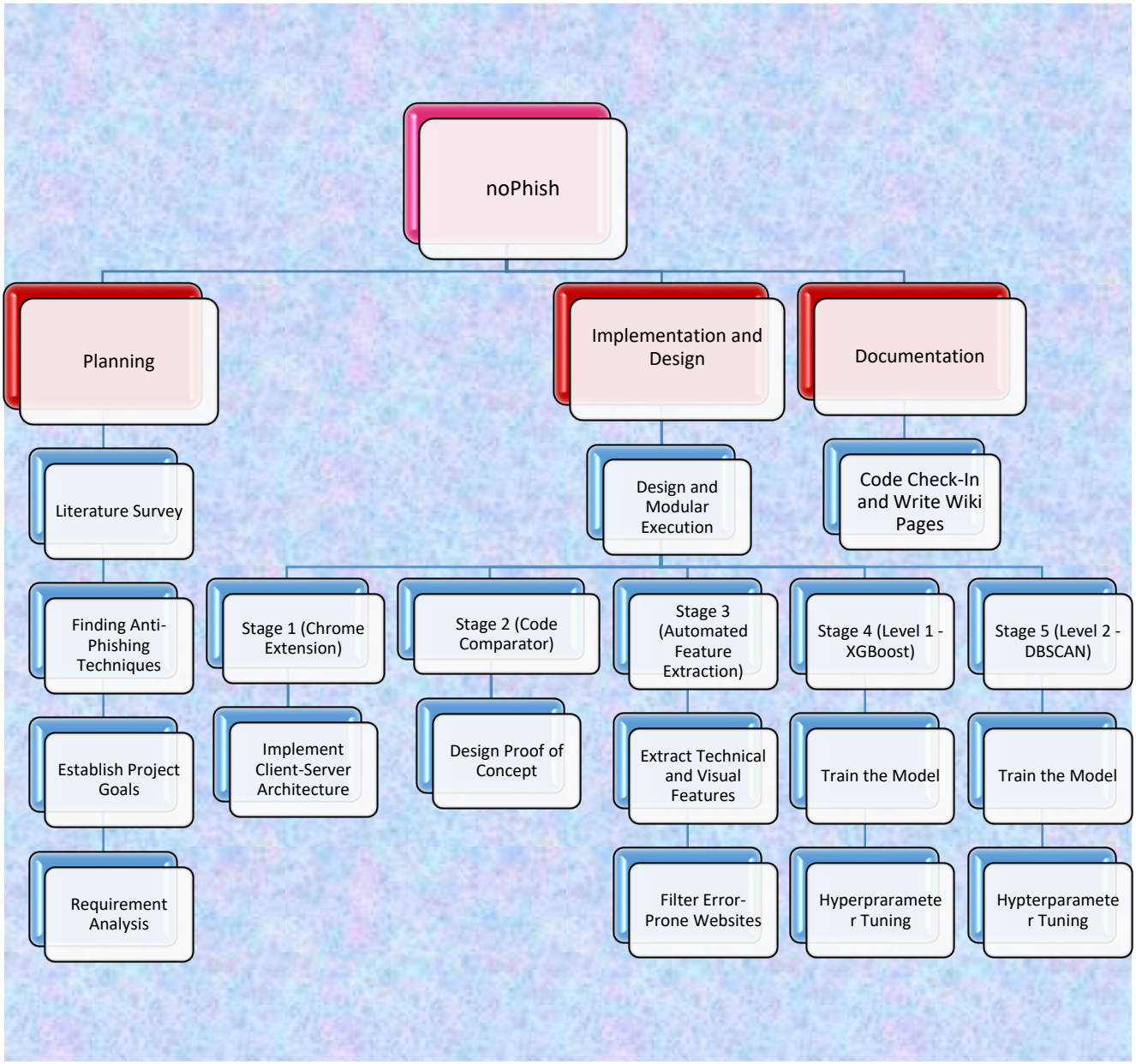
Article XVI. Use Case Diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).



Article XVII. Functional decomposition

Functional Decomposition is the process of taking a complex process and breaking it down into its smaller, simpler parts.



Article XVIII. Work Breakdown Structure and Gantt chart

WBS is a hierarchical and incremental decomposition of the project into phases, deliverables and work packages. There was no formal Work-Breakdown Structure (WBS), but putting the work on a linear timeline, the following WBS and Gantt chart is prepared.

No.	*Task*	*Days*	*Start Date*	*Finish Date*
	Planning	**23**	**10-01-2018**	**02-02-2018**
1.	Literature Survey	08	10-01-2018	18-01-2018
2.	Find Anti-Phishing Techniques	11	18-01-2018	29-01-2018
3.	Establish the Project Goals	02	29-01-2018	31-01-2018
4.	Requirement Analysis	02	31-01-2018	02-02-2018
	Implementation and Coding	**130**	**02-02-2018**	**12-06-2018**
5.	Stage 1 (Chrome Extension)	34	02-02-2018	08-03-2018
6.	Stage 2 (Code Comparator)	16	08-03-2018	24-03-2018
7.	Stage 3 (Automated Feature Extractor)	32	24-03-2018	25-04-2018
8.	Stage 4 (Level 1 – XGBoost)	26	25-04-2018	21-05-2018
9.	Stage 5 (Level 2 – DBSCAN)	22	21-05-2018	12-06-2018
	Documentation	**06**	**12-06-2018**	**18-06-2018**
10.	Code Check-In and Write Wiki Pages	04	12-06-2018	16-06-2018

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis.

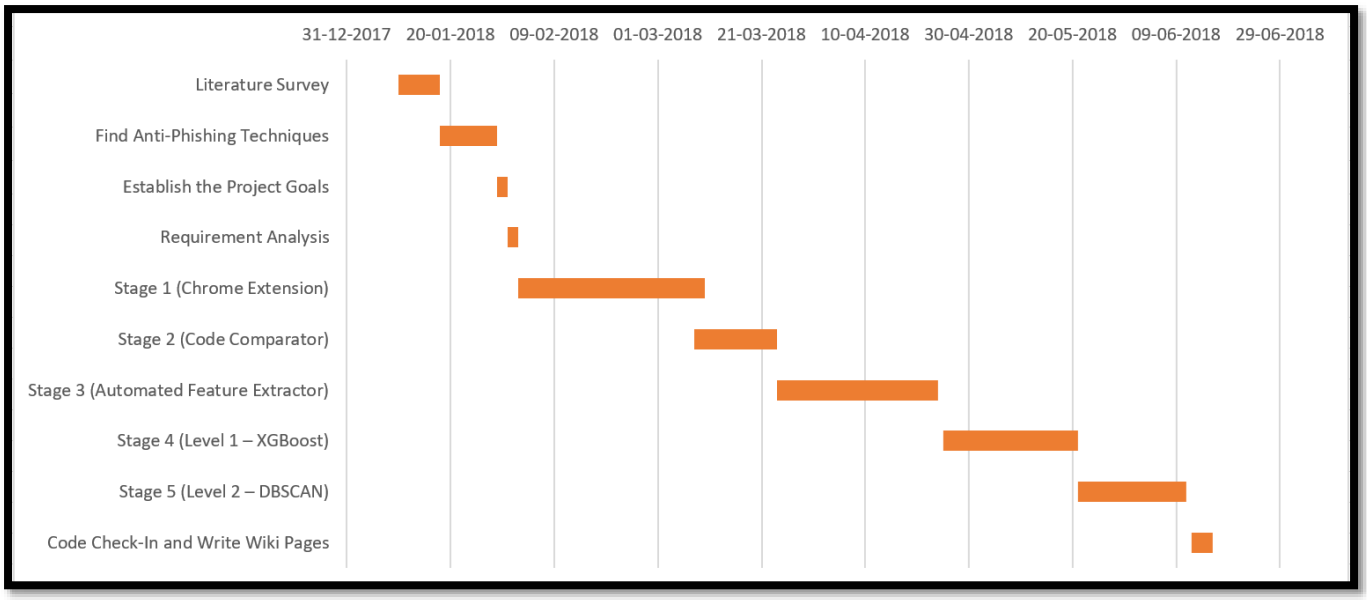


Figure 23. Gantt chart

Article XIX. Conclusion

Malicious URL detection plays a critical role for many cyber security applications, and clearly machine learning approaches are a promising direction. noPhish – an effective method to classify websites into Phishing and Legitimate (Non-Phishing) using Machine Learning, Image Processing, and Data Mining techniques – was successfully built. The methodology exploits not only the technical features of the website based on URL specifics, WHOIS, HTML Headers, SSL/TLS certificate and page source (underlying code) of the website, but also the visual features of the website in real-time. This was done with the integration with Machine Learning techniques which gives it a robust, flexibility and upgradable power as the models can be trained routinely. As a proof of concept, a Chrome browser extension was created which makes a RESTful call to the server where the website's URL will be sent and will receive website's Phishing or Legitimate status in return. It was also established that comparing page source (underlying code of the website) of websites doesn't help in predicting if a website is phishing or not. It is to the best knowledge, that noPhish is the first way to classify website on both the technical and visual aspect of the website in real-time.

Article XX. Further Work

Over the period of this project, a great many improvements and ideas were established that can be investigated in the future.

- Most pages have advertisements on them, i.e. dynamic elements which keep changing and hence, lead to new image hashes being generated every time for the same page. Add a functionality to blank out advertisements from the page before screenshot.
- Ignoring the noise on the webpage and image hashing only the important content can be done. Noise reduction is done by a Chrome Extension called Mercury and its source code can be useful.
- Enable permanent whitelisting of trusted domains that show up as false positives. If a user clicks allow, never show an alert on that website.
- Show the user the precise URL which is being phished.
- Create a database of popular brands' URL and their respective favicon. If any other website uses the same favicon, regard that as phishing. Derive a proof of concept and add it as Level 3 of noPhish.
- More effective machine learning algorithms for training the predictive models.
- More effective feature extraction and representation learning (e.g., via deep learning approaches).

Article XXI. Bibliography

https://www.pinterest.com.au/pin/61220876165007473/visual-search/?x=0&y=0&w=564&h=687
https://www.wired.com/2017/03/phishing-scams-fool-even-tech-nerds-heres-avoid/
http://www.foxnews.com/tech/2017/03/16/warning-dangerous-new-gmail-phishing-attack-can-easily-steal-your-google-login.html
https://www.csoonline.com/article/2132618/phishing/11-tips-to-prevent-phishing.html?upd=1516601104944
https://www.csoonline.com/article/2117843/phishing/what-is-phishing-how-this-cyber-attack-works-and-how-to-prevent-it.html?nsdr=true
https://community.mimecast.com/docs/DOC-1027
https://community.mimecast.com/docs/DOC-1995
ttps://www.mimecast.com/blog/2015/08/under-the-hood-our-new-email-attachment-sandbox/
https://www.sandboxie.com/EmailProtection
https://techterms.com/definition/sandboxing
https://www.fortinet.com/content/dam/fortinet/assets/solution-guides/Why-Do-You-Need-Sandboxing
http://www.kansas.com/news/local/crime/article88960532.html
https://techcrunch.com/2016/03/15/prosecutors-find-that-fappening-celebrity-nudes-leak-was-not-apples-fault/
https://www.cbsnews.com/news/the-phishing-email-that-hacked-the-account-of-john-podesta/