Data Science Analysis of Malicious Advertisements and Threat Detection Automation for Cybersecurity Progress

Sandra Nguyen
Department of Computer Science
California State University, Fullerton
Fullerton, USA
sand.nguyen9@csu.fullerton.edu

Doina Bein
Department of Computer Science
California State University, Fullerton
Fullerton, USA
dbein@fullerton.edu

***Abstract*—We live in an era of unprecedented technology. Millions of users depend on information technology to carry out their daily lives and large-scale commercial and industrial operations are no exception. At the same time, the rapidly growing interconnectivity of IT systems and the surge in cybercrime since the pandemic have rendered industry-standard hardware and software components increasingly vulnerable to malicious attacks. Cyber defense is a coordinated act of resistance that intends to understand the capabilities and motives of attackers in order to secure our country's data and more importantly, the livelihoods of our citizens. This research aims to contribute to the progress of cybersecurity and defense technology as a whole by focusing on a dynamic aspect of malware: digital unwanted advertisements. It presents a novel approach to automating the analysis of malicious content on the internet by web scraping ads of the popular search engine Google to extract relevant data (URL, Company, Title, Product Desc.), building machine learning models (supervised & unsupervised) to classify and make predictions on that data, and creating a web application for end users to access. The results show that our tool can detect trends within the features with limited false positives, paving the way for us to make predictions on whether the advertisements are desirable or unwanted. The research concludes that in this time and age, it is extremely important to protect against fraud, especially by adhering to cybersecurity's best practices and to think about threats in more global terms. Our hope with this research is to prompt action to ensure society continues to improve in IT resilience.**

***Keywords*—machine learning, data science, cybersecurity, Google Ad, malicious URL, shortening service**

I. Introduction

In recent times, online advertising has become the predominant strategy for business marketing, which greatly impacts the day-to-day activities of online users, such as entertainment, communication, banking, and e-commerce. Unfortunately, due to the low cost of digital advertising and lack of user knowledge on installing adware software, online platforms are flooded with unwanted and malignant advertisements that devalue the user experience and deliver complex malicious activities—such as malware, scam, and phishing.

More and more, attackers are targeting unsuspecting users by purchasing Google Adverts based on specific keywords that point to compromised websites. A 2020 investigation [1] carried out by British magazine Which, determined that it was possible to create a credible fake Google ad in just a few hours. With the only requirement being a Gmail address, Journalist Andrew Laughlin managed to promote his fake mineral water brand, Remedii, and accompanying online service, Natural Hydration, with little to no identification or vetting. In the span of a month, the Google ads gained nearly 100,000 impressions for queries such as "bottled water" and "hydration advice."

Therefore, in the absence of digital ad verification, cybercriminals across the world can utilize Google Ads' pay-per-click (PPC) model [2] to trick users into giving up confidential information such as login credentials, account details, and Personal Identifiable Information (PII), which are then used to hijack accounts, drain crypto wallets, and build data points on future victims. A recent malicious phishing attack [3] in early April 2022 targeted Trezor (crypto wallet) users who searched for Trezor-related keywords on Google and were met with ads impersonating the website pop-up. If the victim entered their wallet recovery seed phrase, their crypto wallets would be drained empty by the scammers.

Furthermore, a study on "bad" advertisements [4] by Eric Zeng from the University of Washington concluded that a significant proportion of users fail to recognize certain types of media as ads, which allow common forms of online advertising fraud—phishing for user credentials, stealing money from legitimate advertising campaigns, and/or promoting a fake e-commerce site—to be as lucrative as they are today. Through no fault of their own, a consumer's unfamiliarity with cybersecurity best practices or trusting attitude toward a brand that is subject to data breaches, among other things, may give cybercriminals an unforeseen advantage in today's world.

The exposure of a multimillion-dollar ad fraud scheme [5] by BuzzFeed News in mid-October 2018, may lend us a hand in understanding how these fraudsters evade current, industry-level fraud detection systems. Utilizing the data of real human users on 125 Android apps and websites, they programmed bots to mimic human user behavior and stole close to $10 million from Google's ad networks and partners. This invokes an important question: To what extent has cyberattacks evolved in regards to our ability to detect it? How can we, as a society, improve in cybersecurity vigilance and develop the necessary technologies to remedy this disparity?

Addressing this problem at a fundamental level would require significant changes to the digital advertising ecosystem and education sphere, perhaps ensuring proper implementation of ad verification or allocating resources to students and employees about how to protect their digital assets from vulnerabilities.

Our aim in this paper is more modest, yet it retains the motivation of informing and protecting the individual consumer in the modern online environment. To do this, we must first understand the current, real-world statistical context of malicious activities: the target distribution of benign to malign advertisements; their URLS—which when run through supervised machine learning models with a focus on feature extraction (discussed in more detail below) allows us to make conclusions about what *exactly* constitutes a spoofed URL.

Then, we propose an exploratory detection system for potentially malicious advert URLs, which to be practically useful, has to satisfy three requirements: the data collection has to be *automated* to avoid bias, since Google returns personalized results based on a number of factors; the features have to be *significant* in order for trends in the data to be observed; and the ML models have to be *accurate* in order to produce better predictions.

We achieve this by blending three fields of study, or key methodology aspects:

1. **Data Science Aspect:** Built a web scraper to scrape potentially malicious advertisements on popular search engine Google, using BeautifulSoup, a parsing library, based on user-inputted keywords. Extract relevant data (URL, Company, Title, and Product Description) to a csv file for data analysis by inspecting each search query's web page component. Data is structured with a dictionary so that for every keyword, the top-performing companies will be recorded along with the number of times it appears at the top.
2. **Machine Learning-Focused Aspect:** Employ an end-to-end Scikit-learn workflow: (1) Getting the data ready — cross-validation based on a 70-30% train/test split, (2) Choosing the right ML algorithms, (3) Handling NaN & categorical data — preprocessing, (4) Fitting the models & making predictions, (5) Evaluating the models — accuracy score & silhouette score, and (6) Displaying the results with plots.

   We examine two datasets, an extremely large collection of 651,191 URLS with the classification "benign," "defacement," "phishing," and "malware" published on Kaggle in 2021 [6], and our own 614-count, scraped dataset with an assortment of sample key words, including "nft," "cloud computing services," and "artificial intelligence software."

   Feature engineering includes extracting three types of features: *Lexical*, such as URL length, number of digits & letter characters, symbols (@, ?, #, %, and more), the presence of an HTTPS and shortening service; *Host-Based*, such as domain name; and *Content*, such as having an IP address.

   All supervised & unsupervised machine learning algorithms, including but not limited to Random Forest Classification and K-means Clustering, are built and evaluated according to their accuracy, or silhouette score, to predict and identify patterns.
3. **Web Development Aspect:** Develop a user-friendly web application to allow users to input certain keywords and the number of times to scrape into the ML model in order to receive an unbiased sample. Users will have the ability to display and view their results through the data structure.

To motivate our approach more succinctly, we break down the structure of a URL in *Section II* and discuss manipulation techniques that effectively deter human detection. *Section III* provides details on our web scraping algorithm (including the random sampling technique) and supervised learning approach, along with the results of our feature engineering. We evaluate our exploratory detection system in *Section IV* with a large-scale experiment on our own scraped dataset and demonstrate its ability to detect anomalies in the data. Lastly, we discuss the results in *Section V* and outline conclusions and future directions in *Section VI.*

The significance of our contributions is as follows:

- We present an exploratory detection system for detecting malicious advertisement URLs that uses an automated technique of data collection.
- Our feature engineering procedure returns significant results, which allow our supervised learning approach to exceed expectations; among the three classifiers we consider, Random Forest performs the best (91.4% accuracy), though the others contribute sufficient findings.
- Our unsupervised anomaly detection approach and best-performing K-means model (with a 0.447 silhouette score) suggests interesting relationships between a malicious URL and certain features.

We strive to put power back in online users' hands and allow them to understand the media they interact with day-to-day. We work toward a future in cybersecurity where malicious activity is detected and curbed before it has a chance to inflict damage—a future where AI will aid humans in the pursuit of automating Internet safety.

II. Background

A. Breaking Down the Structure of a URL

To set the scene, we discuss the components of a URL and the cybersquatting techniques that render manual human detection of malicious URLs infeasible. This reasoning is applied to our rationale that machine learning-backed systems are necessary to aid the everyday user navigate the online sphere.

The different parts of the URL are listed below [7]:

1. **The protocol:** Describes the way a browser should retrieve information from a web source, e.g., HTTPS, HTTP, which differ by being secure/not secure, & FTP.

2. **The hostname:** Is made of the domain name and subdomain name.
    a. The domain name consists of the second-level domain which is the name of your website & the top-level domain which specifies the type of website, e.g., .com, .edu. This combination is the only *unique* part of the URL.
    b. The subdomain is a specific "zone" inside the website.
3. **The path**: A path on a personal computer to a specific file, indicating what resource is obtained from the website.

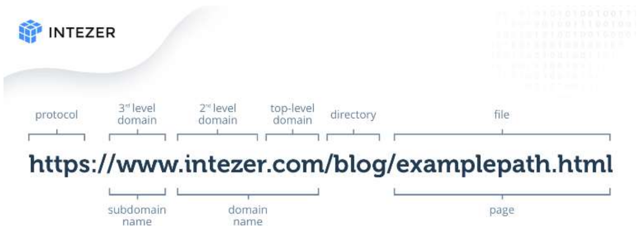


Fig. 1. URL structure

The correct way to read a URL is from *right to left* starting with the hostname. The key to identifying a malicious URL is its unique domain name, which takes us to the actual website. Consider two URLs:

ftp://ftp.microsoft.com/software/patches/fixit.exe

Here, the domain or website is Microsoft.com and the protocol is FTP. This URL is legitimate.

http://securitycenter.microsoft.us.admin-mcas-gov.ms

In this case, it would be ill-advised to assume the word "microsoft" inside the URL proves its legitimacy. The actual website is admin-mcas-gov.ms with the second-level domain being "admin-mcas-gov" and the top-level domain being "ms." This is a phishing URL.

B. *Hacking the Human Brain*

Let's review some manipulation techniques that scammers implement when *cybersquatting*—which refers to the unauthorized registration and use of Internet domain names that are identical or similar to trademarks, service marks, company names, or personal names.

1. **Subdomain Cybersquatting:** This technique targets users' habit of reading English from left to right. As they encounter a recognizable brand name, they tend to immediately trust the URL.
2. **TLD (Top Level Domain) Cybersquatting:** This technique involves mimicking a company's website URL in whole save for the top-level domain. Customers are tricked into believing the copy is the real deal.
3. **Typosquatting:** A basic technique that counts on users making typos when writing popular domain names, e.g., yutube, goggle, micosoft. Similarly, *similar-domain cybersquatting* focuses on the similarity of the domain, e.g., cnn-news.com (added "-news") or Facebook1.com (added digit "1").
4. **Suggestive words**: Words related to inputting credentials for some service, e.g., "login," "bank," and "activate," are likely to be used in phishing URLs.
5. **Short URLs**: This technique hides the true URL by using a short redirect URL, e.g., tinyurl, bit.ly.

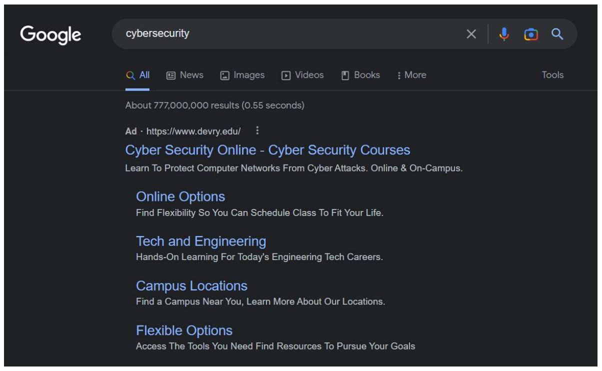
C. *Malvertising vs. Google Ad Scams*

Malvertising, or malicious advertising, [8] is known widely as a cyberattack in which perpetrators inject malicious code into legitimate digital ads. Attackers may perform the following when a user views the advert with or without clicking on it:

1. A "drive-by download" where malware is installed on the computer, made possible due to browser vulnerabilities.
2. Forced redirect of the browser to a malicious site; may be operated by an attacker to carry out a phishing scam.
3. Display of unwanted advertising, malicious content, or pop-ups via JavaScript.

For the purpose of this paper, we will only consider fake Google advertisements [9] that are bought by fraudsters for a variety of reasons:

1. **Cryptocurrency Phishing Scams:** Scammers use domain spoofing (aka cybersquatting) with similar URLs to impersonate well-known digital crypto wallet brands like Phantom and MetaMask.
2. **Brand Impersonation:** Scammers pass themselves off as existing brands by stealing product photos and other intellectual property that appear on their Google Ads.
3. **Spoof Websites:** Scammers take brand impersonation one step further by replicating an entire brand's website.
4. **Counterfeiting:** Scammers can use fake Google ads to direct consumers to websites that sell counterfeit products, which usually reflect poorly on the real brand.



About 777,000,000 results (0.55 seconds)

Ad · https://www.devry.edu/

Cyber Security Online - Cyber Security Courses

Learn To Protect Computer Networks From Cyber Attacks. Online & On-Campus.

Online Options

Find Flexibility So You Can Schedule Class To Fit Your Life.

Tech and Engineering

Hands-On Learning For Today's Engineering Tech Careers.

Campus Locations

Find a Campus Near You, Learn More About Our Locations.

Flexible Options

Access The Tools You Need Find Resources To Pursue Your Goals

Fig. 2. Search campaign ad example for keyword "cybersecurity"

Let's take a step back and rationalize our research approach and goals in this new context. The greatest tool we have in predicting malevolence, without a doubt, is a website or advert's URL—where a conscious human eye can set to uncovering a scammer's handiwork. However, in 2022 alone, the total cost of ad fraud for businesses and consumers was $81 billion [10]. This number is predicted to increase to $100 billion by 2023 due to the never-ending advancement in technology. Organizations and individuals without advanced fraud prevention solutions or skills will be exposed to attacks around the globe. Our mission, simply, is to inform the everyday user of how to safeguard their digital assets, to remind businesses to keep their staff trained and up to date in order to protect their customers, and to remind society to remain vigilant of everyday processes—even something seemingly safe as Google Ads.

III. Methods

A. *Web Scraping Algorithm*

The Google Ads URL web scraper was developed to automatically and repeatedly collect Google advertisements based on a list of predetermined keywords. This list, included below, was compiled through a surveying process amongst peers; with the conditions to providing a "keyword" indicated on the Google Form as "tech" or "product-like".

"nft," "cloud computing services," "malware," "app development," "cryptocurrency," "web design," "web development," "resume building," "infographic maker," "job search sites," "job search," "internship search," "online news," "academic poster template," "video editing software," "photo editing software," "color correction software," "photoshop software," "augmented reality software," "artificial intelligence software," and "adobe software."

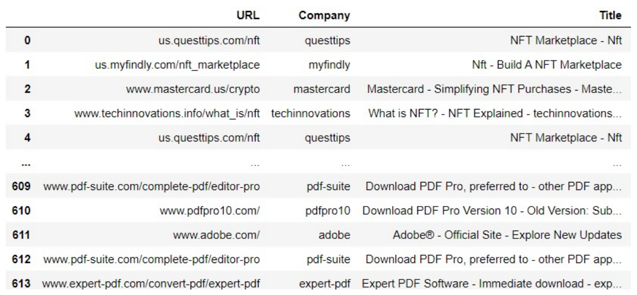
The overall workflow of the Google Ads URL static web scraper contains eight steps (refer to GitHub for full documentation [11]):

1. Import libraries *requests*, *json*, *tldextract*, *bs4*, & *csv*.
2. Specify a user agent [12] to mitigate bot-blocking.
3. Fetch the HTML code for entire Google search query web page with the *Requests* library; the get() method is used to send a GET request to the selected data source. The payload is our targeted keyword.
4. Write source code to HTML file.
5. Check valid request & feed the fetched HTML to *BeautifulSoup* [13] object.
6. Scrape both top and bottom advertisements & extract relevant data (Title, URL, Company, & Product description)—utilizing Chrome's Inspect Element mode (Ctrl + Shift + I) will aid in the process of finding the unique JavaScript class ID. The find() method finds the first tag with the specified id. Implement try-except functions to avoid crashing the program.
7. Write data elements to a CSV file with the *Csv* library.
8. Loop n number of times to obtain more accurate data.

For further keyword and competitor analysis, which is implemented in the web app, refer to the two steps below:

9. Determine absolute-top advertisement company—to elaborate, Google ads claim a top spot (ads positioned above search results) by meeting the Ad Rank thresholds [14], a combination of bid, auction time, and quality.
10. Display to nested dictionary: total absolute-top ads & number of search result ads for a certain company—for a certain keyword.

We ran our Python script in Visual Studio Code on our list of 21 tech-related keywords, for a total of 10 replications—meaning our web scraper fetched the search results of each keyword 10 times; we expect this repetition will account for duplicate advertisements *and* present anomalies that may be our target malicious ads. The program had a run-time of approximately 1-2 minutes, however, was capable of producing a 614-count "scraped" dataset that was later applied to our unsupervised ML models.



	URL	Company	Title
0	us.questtips.com/nft	questtips	NFT Marketplace - Nft
1	us.myfindly.com/nft_marketplace	myfindly	Nft - Build A NFT Marketplace
2	www.mastercard.us/crypto	mastercard	Mastercard - Simplifying NFT Purchases - Maste...
3	www.techinnovations.info/what_is/nft	techinnovations	What is NFT? - NFT Explained - techinnovations...
4	us.questtips.com/nft	questtips	NFT Marketplace - Nft
...	...	...	...
609	www.pdf-suite.com/complete-pdf/editor-pro	pdf-suite	Download PDF Pro, preferred to - other PDF app...
610	www.pdfpro10.com/	pdfpro10	Download PDF Pro Version 10 - Old Version: Sub...
611	www.adobe.com/	adobe	Adobe® - Official Site - Explore New Updates
612	www.pdf-suite.com/complete-pdf/editor-pro	pdf-suite	Download PDF Pro, preferred to - other PDF app...
613	www.expert-pdf.com/convert-pdf/expert-pdf	expert-pdf	Expert PDF Software - Immediate download - exp...

Fig. 3. Scraped dataset

B. *Supervised Learning Approach*

Human beings are always on the search for tools and techniques that reduce the effort of performing a task efficiently. In Machine Learning, algorithms are designed to learn by themselves using past experience, to be capable of reacting and responding to new conditions. When it comes to fraud detection, ML can identify hidden patterns that have not been previously recognized and parallel the skill of today's cybercriminals.

Our choice of classifiers is determined by the corpus of labeled data available online. For the purpose of this study, we examined an extremely large collection of 651,191 URLs, out of which 428,103 are benign/safe URLs, 96,457 are defacement URLs, 94,111 are phishing URLs, and 32,520 are malware URLs. This archive was published to Kaggle [6] by Manu Siddhartha in 2021, incorporating several existing URL datasets, and consists of two columns: URL & Type (benign, defacement, phishing, or malware).

Since the goal of our exploratory detection system is to highlight hidden patterns in the data amongst certain URL features and detect anomalies, it is essential to understand the real-world statistical context of these malicious activities.

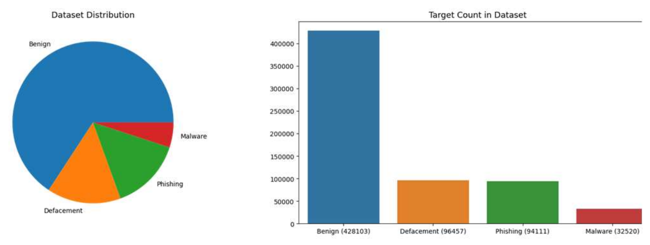


Fig. 4. Target distribution of Kaggle dataset

The End-to-End Scikit-Learn Workflow of our Supervised Learning approach contains eight steps (refer to GitHub for full documentation [11]):

1. Import libraries *pandas*, *numpy*, *preprocessing*, *LogisticRegression*, *RandomForestClassifier*, *DecisionTreeClassifier*, *plt*, *sns*, *accuracy_score*, & *confusion matrix*, among others.

2. Read Kaggle malicious URLs CSV file into Pandas [15] dataframe with the read_csv() method & display head.

3. Data preprocessing: Handling NaN (missing data) with the isnull() method, Omitting (www.) with the replace() method, & Extracting features (detailed below):

Our feature engineering [16] approach involves extracting lexical, host-based, and content-based features from the labeled URL strings in order to identify its malevolence. Definitions and justifications are provided for all three categories:

- **Lexical features:** These refer to statistical features extracted from the literal string. The motivation for including lexical features is based on the fact that malicious URLs have a different appearance to benign URLs, therefore, we can extract statistical properties that quantify this difference. Features chosen are:
    - URL Length, Symbols ('@', '?', '-', '=', '.', '#', '%', '+', '$', '!', '*', ',', '//'), Presence of HTTPS, Digit Count, Letter Count, Presence of Shortening Service, & Presence of IP Address

- **Host-based features:** These are obtained from the host-name properties of the URL and provide information about the host of the webpage, e.g., country of registration, open ports, etc. They can demonstrate a certain characteristics that malicious and benign sites have a reputation of having. Features chosen are:
    - *Domain & Abnormal URL*

- **Content-based features:** These are obtained from the webpage's HTML code, which capture the structure and content embedded in it. Similar to host-based features, they capture characteristics found in compromised pages. We did not consider any content-based features.

In analyzing the results from our preliminary feature extraction, we have made small but startling conclusions:

- Malicious URLs are generally shorter than benign URLs.

- Malicious URLs have an average of at least two periods, while benign URLs have at least one period in them (as expected).

- Malicious URLs contain certain "red flag" or suggestive keywords, including the names of a legitimate company they are targeting.

These findings are consistent with other peer-reviewed conference papers [17] [18] on the topic of feature extraction for phishing URLs and npm packages.

However, contrary to expectations, the feature "presence of a shortening service" did not yield any significant or noticeable results (present in approximately 40,000 URLS or 6%). A *URL shortener* [19] is a website or plugin that is designed to reduce lengthy and complex links. Today, they are popular for social media applications like Twitter which only allow up to 140 characters in a tweet.

URL shorteners work in a sequence of steps:

1. The user inserts a lengthy link into a link shortener website/plugin. The link gets sent to a server for validation.

2. The server inserts the link into a database and generates a shortcode (identification code) in the form of a short link.

3. When a user inserts a shortened URL into a web browser, the server receives the request and retrieves the original URL.

It is worth emphasizing that while a shortening service can offer numerous benefits, scammers and hackers will abuse them to redirect users to look-a-like websites or install malware onto their victims' personal devices. Even LinkedIn has a feature [20] that automatically shortens URLs that are longer than 26 characters; a shortened link will start with "lnkd.in" followed by a random string of characters.

It is challenging to determine where the web browser will take you if the unique domain name is stripped away. For this reason, the "presence of a shortening service" feature became our focused area of interest.

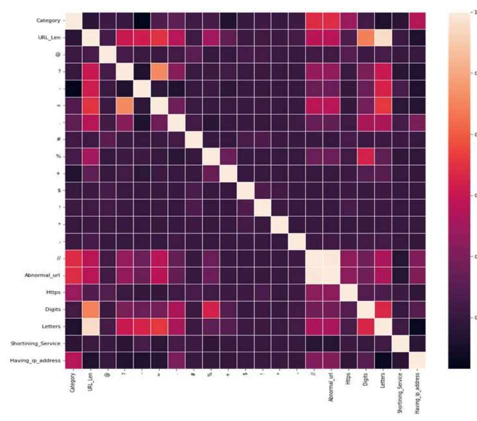


Fig. 5. Seaborn heatmap of features (values near 1.0 are positively correlated)

Continuing our workflow:

4. Getting the data ready — Set 'X' equal to the URL 'Type' object and 'Y' equal to 'Category' multiclass which defines "benign" as '0', "defacement" as '1', "phishing" as '2', and "malware" as '3'.

   Perform cross-validation based on a 70-30% train/test split [21] with the train_test_split() method. This allows us to train our classifiers using a training set of 520,952 URLs and a testing set of 130,239 URLs.

5. We sought learning algorithms that could: (1) Handle imbalanced data well due to the small proportions of malicious data available, and (2) Accurately classify/predict an input into its category. In the end, the ML algorithms [22] that satisfied the constrains were:

   a. **Logistic Regression:** Used to determine if an input belongs to a certain group or not.

   b. **Decision Trees:** Classifiers that are used to determine what category an input falls into by traversing the leaves and nodes of a tree.

   c. **Random Forest:** A collection of many decision trees from random subsets in the data, resulting in a combination of trees that may be more accurate in prediction.

6. Fitting the models on the training sets & making predictions on the testing sets.

7. Evaluating the models — Calculate accuracy score with the accuracy_score() method. The Logistic Regression, Decision Trees, & Random Forest models have the accuracy scores of *79.7%*, *90.8%*, & *91.4%*, respectively. To increase our models' accuracies, we can tune the hyperparameters and repeat the process until we achieve the desired performance.

8. Displaying the results with a Seaborn heatmap [23] for each model's confusion matrix. Add labels (Predicted Classification & Actual Classification), and data count. Our best-performing model has a large percentage of its values in the true negatives (TN) & true positives (TP) region—denoted by the diagonal section.

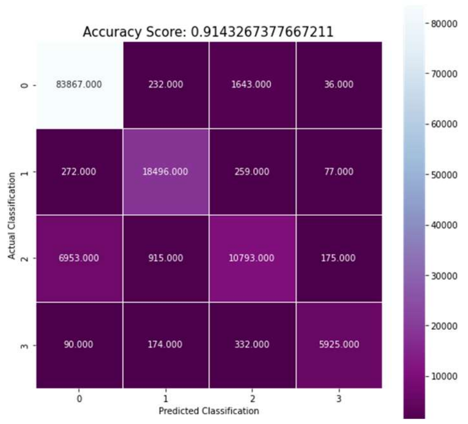


Fig. 6. Best-performing Random Forest Seaborn heatmap

To contrive excellent results in the latter half of our research, we must first understand the real-world distribution of benign to malign URLs. We have taken a deep dive into the statistics—out of a large collection of 651,191 URLs, *34.3%* are malicious (a surprising amount), and confirmed the power of today's leading machine learning algorithms. We have no doubt that AI and ML will continue to disrupt and transform every single segment of society.

IV. Evaluation

A. Unsupervised Learning Approach

While motivating and presenting the details of our approach above, we have argued that its design makes it practically useful in terms of automation, significance, and accuracy. Now, we will support these claims with an experimental study that aims to answer the following research questions:

1. Does our exploratory detection system automate and detect a wide range of malicious advertisement URLs?

2. Does it detect significant patterns and anomalies based on the shortening service feature?

3. Are clustering results cohesive and separated (validation metrics) enough to be useful?

To answer these questions, we conducted a large-scale experiment on our 614-count "scraped" Google Ad URLs dataset to assess performance and accuracy. The End-to-End Scikit-Learn Workflow of our Unsupervised Learning approach contains eight steps—similar to Supervised Learning, with minor tweaks (refer to GitHub for full documentation):

1. Import libraries *pandas*, *numpy*, *preprocessing*, *LabelEncoder*, *StandardScaler*, *KMeans*, *GaussianMixture*, *PCA*, *plt*, *sns*, *silhouette_score*, & *confusion_matrix*, among others.

2. Read Scraped URLs CSV file into Pandas dataframe with the read_csv() method & display head.

3. Data preprocessing: Handling NaN (missing data) with the isnull() method, Omitting (www.) with the replace() method, & Extracting features (detailed above in *Section III B*).

   In analyzing the single feature, "presence of a shortening service", we concluded that our tiny sample of 614 URLs (not even 0.01% of the full 651,191 URLs sample studied earlier), was a *close-to-ideal* representation of the other. How? We performed a mathematical calculation on the distribution of URLs with and without a shortening service and found that approximately 75 URLS or 12% of the data contained the service. According to the principles of statistics [24], larger samples are a closer approximation of the real-world population, which accounts for the discrepancy. These findings were fascinating.

4. Getting the data ready — Set 'X' equal to relevant features, "URL," "Company," "Title," "URL_Len," "Letters," and "Shortening_Service".

Perform *Label Encoding* (scikit-learn library) [25] on categorical features (URL, Company, Title, & Product_Description), which assigns each label to a unique integer based on alphabetical ordering; preferred over One-Hot Encoding due to its high performance on a large number of categories).

Perform a *Standard Scalar* [26] transformation on all categories to standardize, i.e. μ = 0 and σ = 1 our features, individually; to make the data the same scale and dimension. This procedure is recommended after label encoding.

5. We sought learning algorithms that could: (1) Handle imbalanced data well due to the small proportions of malicious data available, and (2) Accurately classify/predict an input into its category. In the end, the ML algorithms that satisfied the constrains were:

   a. **K-Means:** Finds similarities between objects and groups them into K different clusters.

   b. **Gaussian Mixture:** Assumes all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

6. For K-Means (only), find distribution of raw data & apply *Dimensionality Reduction* [27] with Principal Component Analysis (PCA)—which will reduce the dimensionality of a dataset that contains a large number of variables, while retaining as much variance as possible. Reduce to six variables (URL, Company, Title, URL_Len, Letters, and Shortening_Service) & plot seperable data.

7. Find optimal number of clusters via the *Elbow method* [28]—used to find the "elbow" point where adding additional data samples does not change cluster membership; vs. *Silhouette Score*—determines large gaps between each sample within one cluster and across different clusters. Plotting with plt (matplotlib library) aids in the process of choosing 'k' number of clusters.

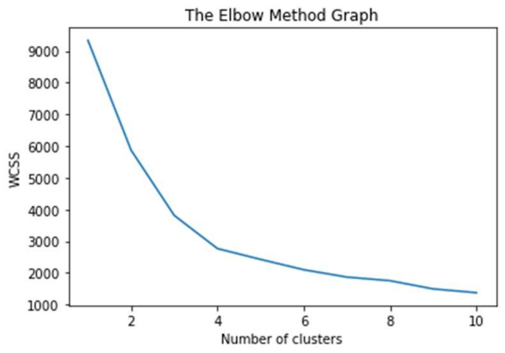


Fig. 7. K-Means Elbow Method graph

- K-Means Elbow method finds k = 4 while Silhouette analysis finds k = 10. We choose to initialize 4 random clusters—along the variables "URL" and "Shortening_Service".

Iterate through each centroid and data point, calculate the distance between them, and assign to a significant cluster [29]. Once the difference between the previously-defined centroids and current ones reach zero, stop.

Visualize the clusters once more.

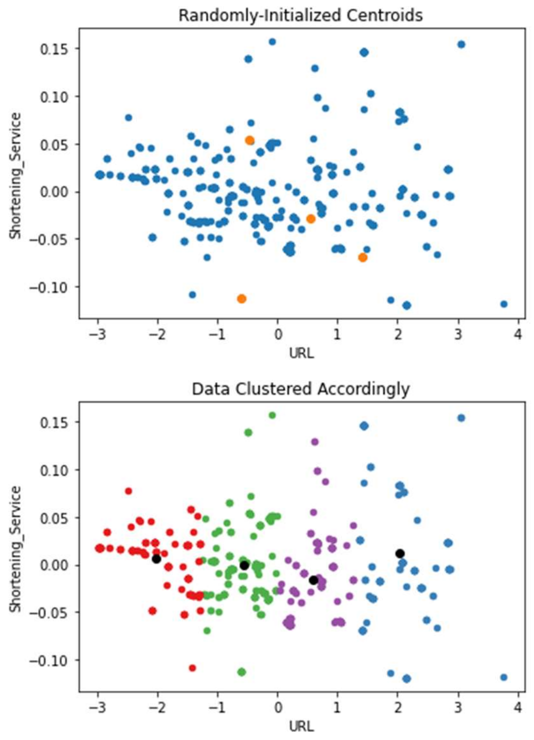


Fig. 8. K-Means clustering technique

- Gaussian Mixture Elbow method finds k = 10 while Silhouette analysis finds k = 5 based on the lowest BIC score.

8. Fit the models on the data, with the optimal number of clusters.

9. Evaluate the models — Calculate silhouette score with the silhouette_score() method. This evaluation metric has a range of 1 to -1; values close to zero indicate that data points are overlapping between clusters. The K-Means and Gaussian Mixture models have the silhouette scores of *0.45* and *0.42*, respectively.

10. Display the results with a Seaborn scatterplot for each model. Add x & y (URL & Shortening_Service), and cluster labels. Our best-performing model denotes high **cohesion** (similarity between the members of a group) within each cluster and **separation** (difference in groups) between the well-defined clusters. (Hint: Think of soccer player uniforms—cohesion is represented by a red jersey for each player in Team A, while separation is achieved if Team A wears red and Team B wears blue on the field.)

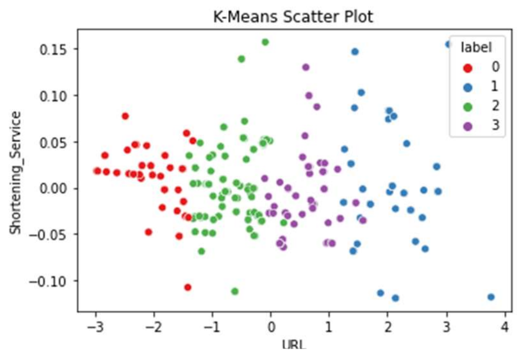


Fig. 9. Best-performing K-Means Seaborn scatterplot

Let's briefly cover the basics of *Anomaly Detection* [30] and quantify the value of our findings with real-world data. **Anomalies**, or outliers, are data patterns that deviate from the *standard* behavior of the rest of the data. To identify them, we use an excellent metric called *Modified Z*-score (zmod), which is the distance between each cluster's point and their respective cluster's centroid, and determine a threshold (>3). Across all 4 K-means clusters, URLs that have a zmod greater than 3 can be linked with having a shortening service present—represented by the shortening_service values being > 0.15 or < -0.10. The visual representation of the points as outliers (approximately *6%*) further confirms this.

Two cases from our scraped dataset are given which highlight the precision of our detection approach:

www.opensea.io/nft-worlds/collection

A manual check of the URL structure and website destination leads us to believe our learning model correctly predicted its benevolence.

business.linkedin.com/linkedin/jobs

This suspicious URL reveals a 404 Page Not Found error and contains a subdomain registered as "business" as well as a misleading 2nd level domain "linkedin". Our model tagged this URL as an anomaly, and it may as well be a scammer's discontinued attempt at imitating the world's leading professional networking site.

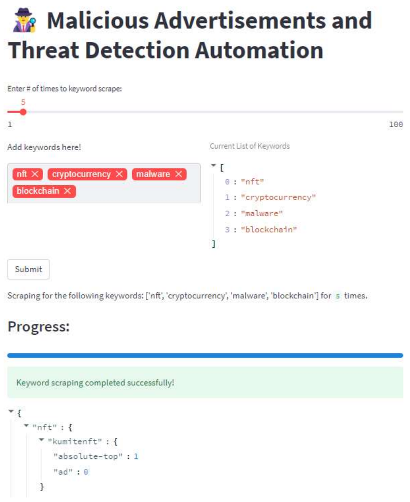
In summary, we conducted an experiment to study the practicality of our exploratory detection system in terms of automation, significance, and accuracy. In our initial trials of building our unsupervised ML models in Jupyter Notebook, we ran our web scraper bot many times and checked the resulting dataset for quality and singularity; while our list of keywords typically concerned themselves with a complimentary list of dominating companies, the data varied in size. Hence, we can answer RQ1 in the affirmative: our exploratory detection system automates and detects a wide range of malicious advertisement URLs. Furthermore, as discussed above, our K-Means clustering model created four well-defined clusters with a silhouette score of 0.45, and detects significant anomalies based on the shortening service feature. Based on these findings, we can give a cautiously positive answer to RQ2 and RQ3.

B. *User-Friendly Web Application*

The interactive Google Ads-scraping keyword analysis web application was developed to aid end-users in analyzing competitors for keywords of their choice. Users have the capability to input keywords and select the number of times the list is scraped by our aforementioned web scraper. The resulting statistics including the advert's "company", "absolute-top ad", and "ad-count" associated with each company, as well as "top performers" and "total-ad-count" for each keyword is displayed through the data structure.

The overall workflow of the Google Ads-scraping keyword analysis web app contains five steps (refer to GitHub for full documentation):

1. Import libraries *streamlit*, *pandas*, *streamlit_tags*, + all included in *Section III A*.
2. Build Title component with the titlte() method, User slider widget (to specify number of times keyword scraping is run) with slider(), and Keywords list for suggested inputs and user inputs appended via st_tags().
3. Build Submit button to call adScraper function and display dictionary with button() method.
4. Define function to house the algorithm of web scraper along with success message, "Keyword scraping completed successfully!" and dictionary data structure. Include Progress Bar initialized to 0 which updates by 0.5 * two parameters after each search query iteration. Use the subheader() and progress() methods.
5. Define new function to turn Json to Pandas data frame.



Malicious Advertisements and Threat Detection Automation

Enter # of times to keyword scrape:

5

1 100

Add keywords here!

nft ✕ cryptocurrency ✕ malware ✕ blockchain ✕

Current List of Keywords

```
[
  0 : "nft"
  1 : "cryptocurrency"
  2 : "malware"
  3 : "blockchain"
]
```

Submit

Scraping for the following keywords: ['nft', 'cryptocurrency', 'malware', 'blockchain'] for 5 times.

Progress:

Keyword scraping completed successfully!

```
{
  "nft" : {
    "kumitenft" : {
      "absolute-top" : 1
      "ad" : 0
    }
```

Fig. 10. Web application dashboard

We developed our Keyword Analysis web application with Visual Studio Code and Streamlit [31], an open-source app framework for data science and machine learning implementations—with adaptations from Andrew-FungKinHo on GitHub [32]. Our mission with this is to help the common user understand how different companies compete and utilize the Google Ads platform to promote their services. Significant Inferences can be made about certain industries and up-andcoming businesses by configuring the right selection of keywords. The possibilities are endless.

V. Discussion

In this section we review the types of malicious advertisements our models found, take a closer look at the Google search algorithm, and touch on tweaks to our approach that we investigated but were shown to be unsuccessful.

It is not within the scope of our unsupervised learning approach to classify each anomaly as defacement, phishing, or malware like the Kaggle dataset had; however, we can make a human inference based on our knowledge of existing malicious activity. For example, a URL that contains a shortening service like Case 2 in *Section IV A*, will likely be a phishing scam directed at users that cannot differentiate between the well-known source and the scam. On the other hand, websites that make use of the HTTP protocol instead of HTTPS in their URLs may be subject to a defacement attack from a malicious party as the connection is not secure. By understanding different key features and the weaknesses they project on a brand, we as consumers can determine the authenticity of a digital advertisement.

Next, we will discuss an intriguing discovery, or rather confirmation of a well-known theory within the world of search engines. On Google's Privacy & Terms [33] page, we find a passage that reads, "The location used and stored with your Web & App Activity can come from signals like the device's IP address, your past activity, or from your device, if you've chosen to turn on your device's location settings," and later, a tip about *Location History* to "help advertisers measure how often an online ad campaign helps drive traffic to physical stores or properties." This may explain why on the job search we are directed to companies hiring in our proximity!

The same conclusion is reached within our data science approach. Figure 11 shows a bar graph of the Top 20 Titles Present in our dataset of scraped URLs. Upon analyzing the graph, we see certain location-specific keywords like "Huntington Beach," "UC Irvine", and "KCAL9 News," which indicates that Google had built us a comprehensive user profile to fine tune our search results. Lo and behold, the web scraper script was run on a personal laptop within the SoCal region.

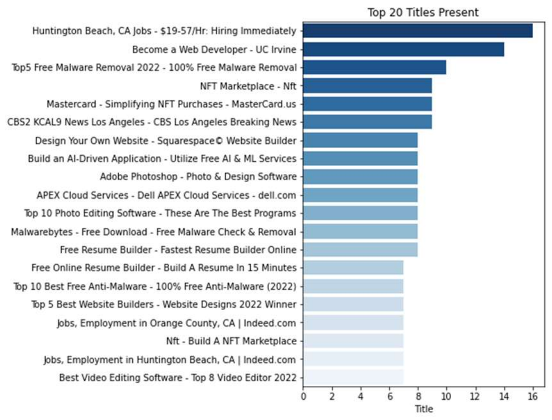


Fig. 11. Top 20 Titles Bar Graph

Lastly, we attempted several tweaks in our code and algorithms that did not prove successful. Literature recommended building a K-Nearest Neighbors model, however, we did not find it to provide an advantage in our experiment. Additionally, we had planned to collect real user data from peers at a scientific conference via our web application, however, without proper back-end implementation on the Cloud, their data could not be stored. We plan to navigate this issue in future works.

VI. Conclusion

Since the mid-1990s, the Internet has developed rapidly and impacted almost if not all aspects of society, for better or for worse. In investigating malware and the relationships between, for example, the surge in cryptocurrency and the rise in cyberattacks, as well as cybersecurity best practices, it led us to a realization that we have reached a point of no return in technology. Our digital footprint, the things we shop for, our ideas and opinions, our locations—everything is tracked and stored in some company's database ready to become the new baseline. As users, our data has become the new gold. And hackers want to steal it.

We have presented a three-prong exploratory detection system to scrape diverse Google Ads data, predict malign URLs and detect anomalies in a supervised and unsupervised ML approach, and help end-users understand the current advertising industry via an interactive web application. Our best-performing classifiers are trained on known samples of malicious and benign URLs and comparatively, unknown real-world Google advert URLs. It works on a set of features extracted using various data engineering techniques such as selecting relevant features, handling missing data, encoding, and normalizing.

We have presented an evaluation of our approach employing three different kinds of classification algorithms and two clustering algos: Logistic Regression, Random Forest, and Decision Tree, as well as K-Means and Gaussian Mixture. In our experiments, all techniques successfully predicted data labels and to a degree of accuracy and detected previously unknown malicious URLs, or anomalies—with Random Forest and K-Means outperforming the others. While all models produced false positives, their precision can be dramatically improved by continually retraining the models. Furthermore, we have shown that our exploratory detection system is automotive, significant, and accurate which suggests it is practically useful.

For future work, we plan to do an extensive investigation on real-world fraudulent digital advert methods employed by scammers and the real process by which search engines are posted to analyze the root cause of the problem. We would like to expand the scope of the data collected from Google Ads to competitor search engines (Yahoo, Bing) which also suffer from tempering and better ensure random sampling via traditional surveying and Google Trends. Another area worth exploring is how to implement the backend via Cloud Computing tools, i.e., AWS Lambda or Microsoft Azure. The frontend web application may be deployed via an Extract-Transform-Load (ETL) process using Apache Airflow. Finally, it would be a great achievement if we could apply our ML techniques to our web application and create an environment where users can validate their own set of advertisement URLs in real-time.

Ultimately, the key to preventing cyberattacks is to remain vigilant of everyday processes—even something seemingly safe as Google Ads may contain weak points that can be targeted by cybercriminals looking for easy money. On the consumer level of society, it is essential to keep users informed and trained to protect their digital assets, while businesses and companies must keep their industry-level systems up to date in order. Trust goes hand in hand with security.

ACKNOWLEDGMENTS

The research presented in this paper is part of the ASSURE-US Summer Research Experiences and is entirely supported by the National Science Foundation for the project Building Capacity: Advancing Student Success in Undergraduate Engineering and Computer Science Award #1832536.

REFERENCES

[1] Andrew Laughlin, Fake ads; real problems: how easy is it to post scam adverts on Facebook and Google?, July 2020, [online] Available: https://www.which.co.uk/news/article/fake-ads-real-problems-how-easy-is-it-to-post-scam-adverts-on-google-and-facebook-aBRVx1e3HVF5

[2] Tony Tran, A Beginner's Guide to Using Google Ads (Previously Google Adwords), September 2020, [online] Available: https://blog.hootsuite.com/google-ads/

[3] Nikhil Panwar, How Scammers Use Google Ads to Target Brands & Customers, May 2022, [online] Available: https://securityboulevard.com/2022/05/how-scammers-use-google-ads-to-target-brands-customers/

[4] Eric Zeng, Tadayoshi Kohno, Franziska Roesner, What Makes a "Bad" Ad? User Perceptions of Problematic Online Advertising, 2021, In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21), Association for Computing Machinery, New York, NY, USA, Article 361, 1-24. https://doi.org/10.1145/3411764.3445459

[5] Craig Silverman, Apps Installed on Millions of Android Phones Tracked User Behavior To Execute A Multimillion-Dollar Ad Fraud Scheme, October 2018, [online] Availabe: https://www.buzzfeednews.com/article/craigsilverman/how-a-massive-ad-fraud-scheme-exploited-android-phones-to

[6] Manu Siddhartha, Malicious URLs dataset, 2022, [online] Available: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset

[7] Daniel Pienica, URL Analysis 101: A Beginner's Guide to Phishing URLs, February 2022, [online] Available: https://www.intezer.com/blog/malware-analysis/url-analysis-phishing-part-1/

[8] Malvertising, December 2019, [online] Available: https://www.imperva.com/learn/application-security/malvertising/

[9] Krynn Hanold, How to report fake Google ads that pretend to be your brand, November 2022, [online] Available: https://www.redpoints.com/blog/report-fake-google-ads/#:~:text=There%20have%20been%20numerous%20reports,they%20are%20the%20real%20company

[10] Artyom Dogtiev, Ad Fraud Statistics, January 2023, [online] Available: https://www.businessofapps.com/ads/ad-fraud/research/ad-fraud-statistics/

[11] Nguyen (2022) data-science-analysis-of-malicious-ads-and-threat-detection-automation-for-cybersecurity-progress [source code], https://github.com/alpacaJin/data-science-analysis-of-malicious-ads-and-threat-detection-automation-for-cybersecurity-progress

[12] How to Fake and Rotate User Agents Using Python 3, January 2023, [online] Available: https://www.scrapehero.com/how-to-fake-and-rotate-user-agents-using-python-3/

[13] Beautiful Soup Documentation, [online] Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[14] Ad Rank thresholds: Definition, [online] Available: https://support.google.com/google-ads/answer/7634668

[15] User Guide#, [online] Available: https://pandas.pydata.org/docs/user_guide/index.html

[16] Ruth Eneyi Ikwu, Extracting Feature Vectors From URL Strings For Malicious URL Detection, Aug 2021, [online] Available: https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

[17] URL Shorteners: Pros and Cons, December 2022, [online] Available: https://easydmarc.com/blog/url-shorteners-pros-and-cons/

[18] A. Sejfia, M. Schafer, Practical Automated Detection of Malicious npm Packages, 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), Pittsburgh, PA, USA, 2022, pp. 1681-1692, https://doi.org/10.1145/3510003.3510104

[19] Ciza Thomas, Detection of phishing URLs using machine learning techniques, December 2013, [online] Available: https://www.researchgate.net/publication/269032183_Detection_of_phishing_URLs_using_machine_learning_techniques

[20] Short URLs in shared posts: LinkedIn Help, [online] Available: https://www.linkedin.com/help/linkedin/answer/a521889/short-urls-in-shared-posts?lang=en

[21] Giorgos Myrianthous, How to Split a Dataset Into Training and Testing Sets with Python, April 2021, [online] Available: https://towardsdatascience.com/how-to-split-a-dataset-into-training-and-testing-sets-b146b1649830#:~:text=The%20simplest%20way%20to%20split,the%20performance%20of%20our%20model

[22] What are Machine Learning Models? December 2022, [online] Available: https://www.databricks.com/glossary/machine-learning-models#:~:text=A%20machine%20learning%20model%20is,sentences%20or%20combinations%20of%20words

[23] Dennis T, Confusion Matrix Visualization, July 2019, [online] Available: https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea\

[24] Lecture4, [online] Available: https://web.pdx.edu/~newsomj/pa551/lecture4.htm#:~:text=Larger%20samples%20more%20closely%20approximate,the%20sample%20size%20is%20large

[25] Alakh Sethi, One-Hot Encoding vs. Label Encoding using Scikit-Learn, March 2020, [online] Available: https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/

[26] sklearn.preprocessing.StandardScaler, [online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[27] Natasha Sharma, K-Means Clustering Explained, November 2022, [online] Available: https://neptune.ai/blog/k-means-clustering#:~:text=K%2Dmeans%20is%20a%20centroid,of%20groups%20in%20the%20dataset. https://towardsdatascience.com/dimension-reduction-techniques-with-python-f36ca7009e5c

[28] Ajitesh Kumar, Elbow Method vs Silhouette Score – Which is Better?, November 2021, [online] Available: https://vitalflux.com/elbow-method-silhouette-score-which-better/#:~:text=The%20elbow%20method%20is%20used,cluster%20or%20across%20different%20clusters

[29] Natasha Sharma, K-Means Clustering Explained, November 2022, [online] Available: https://neptune.ai/blog/k-means-clustering#:~:text=K%2Dmeans%20is%20a%20centroid,of%20groups%20in%20the%20dataset

[30] Issac Arroyo, Unsupervised Anomaly detection on Spotify data: K-Means vs Local Outlier Factor, February 2022, [online] Available: https://towardsdatascience.com/unsupervised-anomaly-detection-on-spotify-data-k-means-vs-local-outlier-factor-f96ae783d7a7

[31] API reference, [online] Available: https://docs.streamlit.io/library/api-reference

[32] FungKinHo (2022) Youtube [source code], https://github.com/Andrew-FungKinHo/YouTube

[33] Google Privacy Policy, How Google uses location information, [online] Available: https://policies.google.com/technologies/location-data?hl=en-US