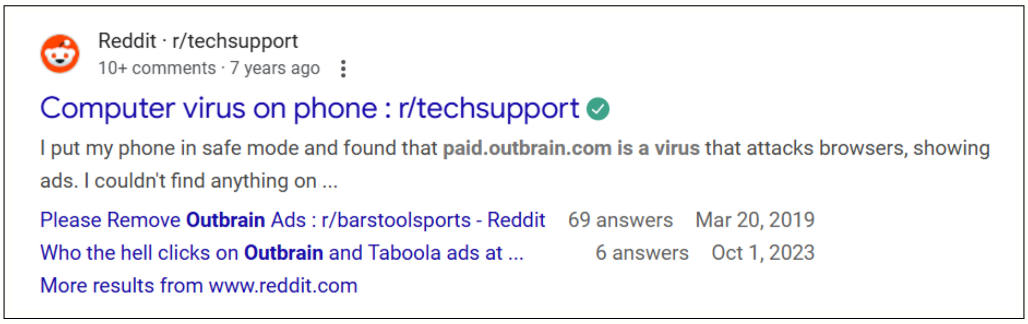
1.3 Cybersecurity Use Case: Detecting Fraudulent Internet Traffic

This section features an anomaly detection agent, part of an LLM to automatically detect abnormal geospatial and temporal patterns with Python code, SQL queries within Python and other techniques to process a repository of Excel spreadsheets.

1.3.1 Introduction: click fraud

In this context, the client is an advertiser on Google and Bing. One of its competitors is suspected to generate fraudulent clicks to gain an unfair and illegal competitive advantage. The data clearly exhibits confirmed click fraud: see the discussion about the hourly spikes in this report. In addition, there is considerable, highly suspicious activity consistent with illegitimate traffic hidden under VPNs, as well as via other mechanisms, with too much of multiples IPs per device in 2019 and 25% concentration in 3 unusual zip-codes (data center related) in 2022. The shift in patterns in 2022 may indicate enhanced tactics to avoid detection, compared to 2019.



Reddit · r/techsupport
10+ comments · 7 years ago

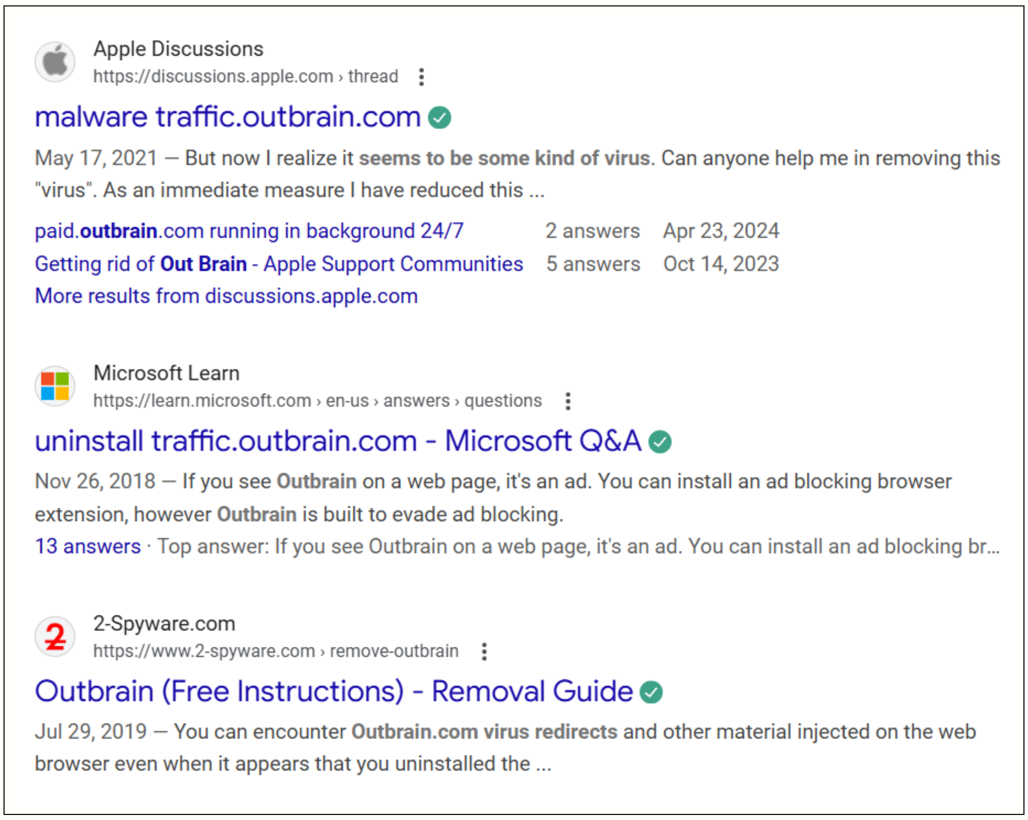
Computer virus on phone : r/techsupport

I put my phone in safe mode and found that **paid.outbrain.com is a virus** that attacks browsers, showing ads. I couldn't find anything on ...

Please Remove **Outbrain** Ads : r/barstoolsports - Reddit	69 answers	Mar 20, 2019
Who the hell clicks on **Outbrain** and Taboola ads at ...	6 answers	Oct 1, 2023

More results from www.reddit.com

Figure 1.6: Google search results about OutBrain (part 1)



Apple Discussions
https://discussions.apple.com › thread

malware traffic.outbrain.com

May 17, 2021 — But now I realize it **seems to be some kind of virus**. Can anyone help me in removing this "virus". As an immediate measure I have reduced this ...

paid.**outbrain**.com running in background 24/7	2 answers	Apr 23, 2024
Getting rid of **Out Brain** - Apple Support Communities	5 answers	Oct 14, 2023

More results from discussions.apple.com

Microsoft Learn
https://learn.microsoft.com › en-us › answers › questions

uninstall traffic.outbrain.com - Microsoft Q&A

Nov 26, 2018 — If you see **Outbrain** on a web page, it's an ad. You can install an ad blocking browser extension, however **Outbrain** is built to evade ad blocking.

13 answers · Top answer: If you see Outbrain on a web page, it's an ad. You can install an ad blocking br...

2-Spyware.com
https://www.2-spyware.com › remove-outbrain

Outbrain (Free Instructions) - Removal Guide

Jul 29, 2019 — You can encounter **Outbrain.com virus redirects** and other material injected on the web browser even when it appears that you uninstalled the ...

Figure 1.7: Google search results about OutBrain (part 1)

There is a significant amount of traffic originating from OutBrain, a company known to use a mix of legitimate activity along with shady tactics to generate paid clicks. Figures 1.6 and 1.7 reveals top Google search results when checking if it is a legitimate company. They may install adware (a form of virus) unknowingly on a large number of devices, typically when the victim downloads freeware, sometimes Trojan anti-malware. The author of this report has experienced similar schemes on his laptop, where the default Google search was changed to a Yahoo partner search infested with paid links from real advertisers.

It results in loss of money for the advertiser due to user (the website visitor) not converting to leads or sales after realizing he was tricked into clicking by click bait and noticing his default search engine was hijacked. Worse, the user believes that the paid links comes from a bad advertiser associated to a fraud scheme, when the advertiser is actually a victim.

The purpose of this study is to assess the amount of fraud that can be attributed to "advertiser fraud". Click fraud shows up in two different versions:

- **Publisher fraud** is perpetrated by a Google affiliate (partner) displaying paid clicks on its websites, for keywords relevant to its audience, that is, relevant to the material published on the websites in question. Typically, the referral field in click logs is not Google nor one of its top partners (YouTube) but instead smaller domain names.
- **Advertiser fraud** aims at depleting the Ad budget of competitors while generating traffic that do not convert to leads or sales. It may involve VPNs and non-static rotating IP addresses to avoid detection by Google, using tools such as BrightData, with multiple IPs attached to a same device. Some red flags may include a large volume of IP addresses that have the exact same small number of clicks each day. The fraud may be perpetrated directly on Google rather than on its affiliates.

Since the perpetrator must generate both impressions and clicks to look legitimate, the bad actor (the competing advertiser) may end up lowering the bid on targeted keywords for the victim due to high CTR (click-through rate) while having the opposite effect on his Ad campaigns. To counter that effect, the bad actor may generate fake clicks against himself to increase his CTR, to appear more relevant to users: in one scheme designed and published by the author of this article, it resulted in all competitors vanishing on Google, leaving the operator as sole advertiser for specific keywords, also lowering bids when competition is gone.

Large-scale sophisticated click fraud targeting specific keywords may have a noticeable impact on the click distribution over time and zipcodes. For instance, traffic related to golfing peaks in June and is minimum in January. Google searches are more common in the South while places like the Northeast or Pacific Northwest receive fewer golf-related queries. If click logs show a strong departure from this natural pattern, it is a red flag.

There are tools available to assess the popularity of keywords, along with trends and geospatial information, to match against what you find in your click logs. Unpopular keywords getting too much traffic is also a red flag. When a click is at the intersection of multiple red flags, it further boost its chance to be fraudulent or undesirable. Poor conversion rates and high CTR attached to web traffic segments are other indicators of problematic traffic within these segments, when far outside the norms. Comparison with non-infected segments or with organic traffic, can confirm fraud.

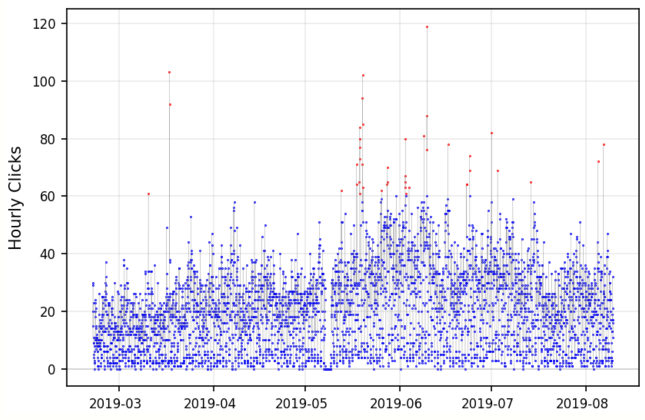


Figure 1.8: Hourly clicks, 2019 – each dot represents one hour (spikes in red)

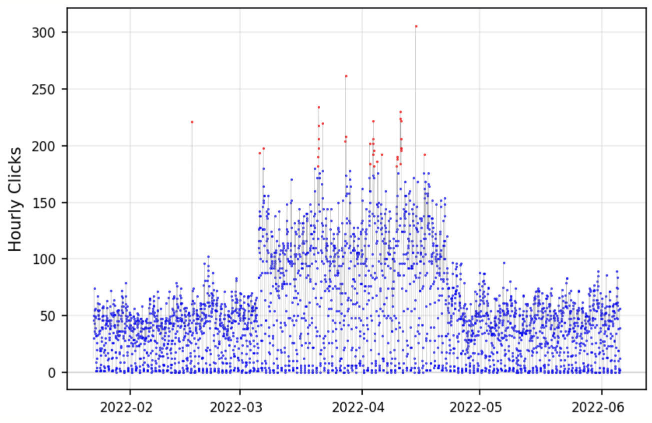


Figure 1.9: Hourly clicks, 2022 – each dot represents one hour (spikes in red)

1.3.2 Click data: general patterns

The data consists of granular click reports spanning across several months in 2019 and 2022. For each paid click, the following fields are available: IP address and ISP along with location, time stamp, landing page URL with click ID, user agent (browser), keywords, device type and device ID. The reports come from the agency managing the keyword campaigns on behalf of the client (the advertiser). The referral URL, bid, CTR and conversions were not included in the preliminary reports shared with me.

1.3.2.1 Time series analysis: random spikes

Figures 1.8 and 1.9 show hourly click totals as time series, over two 5 to 6 months time periods. Blue dots correspond to normal traffic: low hourly click count at night time, higher volume at peak hours, thus explaining why the dots are distributed in two separate bands, visible in the pictures.

Date	Hour	Count	Date	Hour	Count	Date	Hour	Count
2019-03-10	14:00	61	2019-05-19	19:00	85	2019-06-09	19:00	88
2019-03-17	11:00	103	2019-05-19	20:00	102	2019-06-09	20:00	76
2019-03-17	13:00	92	2019-05-19	22:00	63	2019-06-09	21:00	119
2019-05-12	18:00	62	2019-05-26	01:00	62	2019-06-16	18:00	78
2019-05-17	18:00	71	2019-05-27	19:00	64	2019-06-22	19:00	64
2019-05-17	22:00	64	2019-05-27	20:00	65	2019-06-22	21:00	64
2019-05-18	17:00	65	2019-05-27	22:00	70	2019-06-23	20:00	69
2019-05-18	18:00	73	2019-06-02	17:00	67	2019-06-23	21:00	74
2019-05-18	19:00	84	2019-06-02	18:00	65	2019-07-01	01:00	82
2019-05-18	20:00	61	2019-06-02	19:00	63	2019-07-03	00:00	69
2019-05-18	21:00	77	2019-06-02	20:00	80	2019-07-13	21:00	65
2019-05-18	22:00	80	2019-06-02	21:00	61	2019-08-04	23:00	72
2019-05-19	16:00	71	2019-06-04	00:00	63	2019-08-06	17:00	78
2019-05-19	18:00	94	2019-06-08	20:00	81			

Table 1.3: Click counts – hours with largest click spikes (2019)

Date	Hour	Count	Date	Hour	Count	Date	Hour	Count
2022-02-16	16:00	221	2022-03-27	22:00	208	2022-04-09	21:00	188
2022-03-05	21:00	194	2022-04-02	23:00	202	2022-04-09	22:00	190
2022-03-06	21:00	198	2022-04-03	01:00	184	2022-04-10	17:00	230
2022-03-20	18:00	182	2022-04-03	17:00	206	2022-04-10	18:00	184
2022-03-20	19:00	190	2022-04-03	18:00	202	2022-04-10	19:00	224
2022-03-20	20:00	198	2022-04-03	20:00	222	2022-04-10	20:00	198
2022-03-20	21:00	218	2022-04-03	21:00	192	2022-04-10	21:00	222
2022-03-20	22:00	234	2022-04-03	22:00	196	2022-04-10	22:00	196
2022-03-20	23:00	206	2022-04-04	01:00	182	2022-04-10	23:00	198
2022-03-22	00:00	220	2022-04-04	20:00	186	2022-04-11	00:00	206
2022-03-27	18:00	204	2022-04-05	23:00	192	2022-04-14	15:00	306
2022-03-27	19:00	262	2022-04-09	18:00	182	2022-04-16	19:00	192

Table 1.4: Click counts – hours with largest click spikes (2022)

Click counts are higher in 2022, especially in the middle of the time period, when the advertiser probably increased its Ad spend or changed campaign settings. However, the red dots show unexpected spikes happening at random times. These spikes would be harder to detect in charts showing daily totals, but are very visible in hourly time series. Spike details are listed in Tables 6.1 and 1.4. It is interesting to note that **even though spikes appear at random days, they are concentrated on specific hours**: typically between 6pm and 10pm, both in 2019 and 2022. I am not sure which time zone is used for reporting purposes.

zipcode	2019	2022	zipcode	2019	2022	zipcode	2019	2022
02143	314	911	20149	0	942	90001	323	1139
07175	73	1229	22226	71	13093	90189	200	1179
07921	2810	76	31139	993	2646	97501	126	1644
08054	6624	190	33231	455	375	98004	1844	225
10081	654	1210	64468	0	941	98073	3502	1655
10116	426	1867	80259	64	2583	98164	121	20255
15295	135	9034	85001	248	903	77494	0	0

Table 1.5: Top zipcodes for total clicks, 2019 vs 2022 time period

1.3.2.2 Geospatial data: problematic zipcodes

The top zipcodes in terms of clicks are listed in Table 1.5. The number one, zipcode 98164, while in downtown Seattle, has a population of about 200, with too many clicks in 2019, and a gigantic spike in 2022. This is further discussed in section 7.7. By contrast, the largest zipcode in terms of population, US zipcode 77494 with over 100k inhabitants, has zero click, fewer than the expected count (about 100+). Comparing the years 2019 and 2022 shows dramatic shifts.

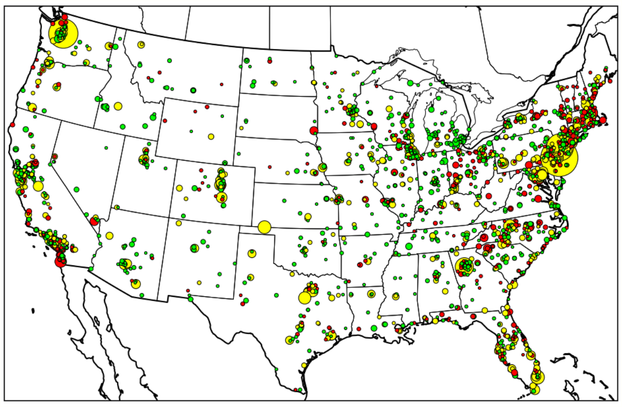


Figure 1.10: Each circle is a zipcode; size represents click volume, color indicates fraud risk (2019)

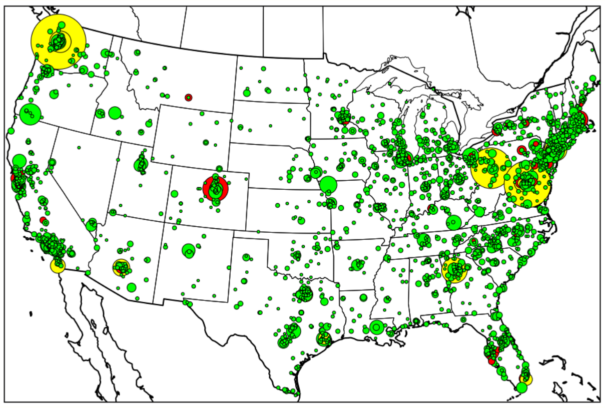


Figure 1.11: Each circle is a zipcode; size represents click volume, color indicates fraud risk (2022)

Figures 1.10 and 1.11 show click locations, respectively in 2019 and 2022, for zipcodes with at least 5 clicks. As in Table 1.5, click numbers (represented by circle size) have doubled between 2019 and 2022. However, locations with clicks growing by a factor above 10 are suspicious, and can indicate the presence of click farms disguised behind data centers. Note that zipcode tracking is not a perfect science.

Each circle represents a zipcode; the circle area represents total clicks during the time period. The color represents suspicious activity, identified as devices with multiple IP addresses. In 2019, when the proportion of clicks from such devices is above 10%, the zipcode is flagged in red. Yellow corresponds to a proportion strictly above 0% but below 10%. Green means no such activity attached to the zipcode in question. If using the same thresholds in 2022, all the zipcodes would be in green. The maximum number of IP addresses per device has been significantly reduced, possibly to avoid detection. In short, **fraud patterns have shifted between 2019 and 2022**, presumably to better avoid detection. For 2019, I only flagged devices with 10+ IP addresses. For 2022, all devices had fewer than 3 IP addresses, and I had to lower my threshold accordingly for flagging purposes. Of course, devices with multiple IPs is just one indicator and by itself, not enough to prove fraud.

1.3.3 Pattern discovery: deeper dive

Here, I illustrate the general framework discussed in section 1.3.2 with case studies. First, I feature explicit examples with detailed insights in sections 1.3.3.1 and 1.3.3.2. Then, I discuss a different type of bad traffic in section 1.3.3.3, this time involving VPNs.

1.3.3.1 Anatomy of an hourly spike

I looked at the details for two spikes occurring on March 17, 2029, respectively at 11am and 1pm. I don't know which time zone is used in the reports, though this traffic is without a doubt, illegitimate. Both spikes are listed in Table 6.1 and visible in Figure 1.8. They are just two among many others. Each may have specific patterns. But these two ones share the same patterns:

- The traffic is very intense during very short time periods (120 seconds) and then remains at a high level for over 40 minutes. At its peak, the velocity is at least 10 times above average. Quite likely, the activity gets detected by Google and the bad traffic source is then blocked or marked as unbillable.
- All the traffic originates from Canada; prior and after the spike, the traffic is mostly US. During the spike, the US traffic remains stable but dwarfed by clicks from Canada.
- The fraudulent clicks originates from a variety of ISPs, IP addresses, browsers and so on, consistent with Botnet activity. Locations are mostly large cities in Canada, predominantly Toronto. The ISPs involved are large ones. The IP addresses are somewhat clustered, but less than you would expect from a classic Botnet, suggesting that the fraud is not coming from an operator that controls its own servers.

This fraud looks like an attack rather than a scheme to defraud Google. It may involve automation or cheap labor paid to do it. The fact that such spikes occur regularly rules out a click bot that went awry, but instead points to a deliberate attempt to hurt the advertiser, and let him know that he is under attack by someone who can hide its origin very well. Professional fraudsters focused on revenue generation tend to be more discrete to avoid detection. Here, it looks like the the bad actor wants to be noticed, while remaining anonymous, as if to discourage the victim from continuing its Ad campaigns.

1.3.3.2 Anatomy of a giant suspicious zipcode

In the 2022 dataset **about 25% of the clicks come from just 3 zipcodes**: 15295 (Pittsburgh, PA), 22226 (Arlington, VA), and 98164 (Seattle, WA) with substantial differences both in volume and location compared to 2019. Most of the IP addresses are attached to the largest ISPs, including Comcast, Verizon, AT&T and so on.

These ISPs show up in the data through a large number of zipcodes across the US. Thus this concentration is peculiar. Note that (say) Comcast IP addresses do not reveal a user's exact zipcode to the general public. It typically indicates only a general location, such as the city, region, or sometimes an approximate zipcode, but the accuracy can vary widely. It would be interesting to obtain the area code of the devices attached to these IP addresses; much of this traffic comes from cell phones, though the information can be spoofed.

As a side note, the author of this article used Bright Data to avoid blocking when crawling large websites. The IP addresses in this network show up under the Internet Service Providers (ISPs) of the actual residential or mobile users who have consented to be part of the network. **Bright Data** does not "own" most of these IPs in the traditional sense; rather, it manages a network that routes user traffic through the IP addresses of real people who have opted into sharing their connection in exchange for a benefit (such as a free app or service), with clear and informed consent. The IP addresses provided by Bright Data fall under several categories:

- **Residential Proxies**: Over 150 million IPs that show up under the ISP of the actual residential internet service provider of the consenting user.
- **Mobile Proxies**: Over 7 million IPs that appear under various mobile network operators/carriers.
- **ISP Proxies**: Over 700,000 dedicated and shared static "ISP proxies" which are data center IPs that are assigned to an ISP, making them appear more like genuine residential IPs.
- **Datacenter Proxies**: Over 1.6 million IPs sourced from data centers, which would appear under the ISP of the data center provider.

The specific ISP that an IP address shows up under will vary widely depending on the type of proxy used and the location targeted. Customers can even target IPs by a specific Autonomous System Number (ASN) or carrier name using the Bright Data control panel. It is thus possible to generate artificial traffic that looks normal, using this service. Yet, in our current case, the extreme concentration in 3 zipcodes looks suspicious, and could suggest that the click fraud scheme is not state-of-the-art and not based on Bright Data.

Another mechanism not suspected here consists of spreading benign Internet viruses (via free downloads offered by a bad operator) to generate clicks from the infected devices. That way, the IP and ISP are both spoofed; it is pretty easy to spoof the browser (user agent). Even if 99% of the clicks are flagged as illegitimate by Google and thus not showing up in click reports, with a large enough volume, the amount of fake traffic can be significant. The operator (the bad actor) can spread viruses that self-kill after generating a pre-specified number of clicks.

To conclude, it would be interesting to find when zipcode 98164 started to generate massive amounts of clicks, and whether it was sudden or not, to rule out a legitimate reason. A quick search about "Why 10% of my clicks come from zipcode 98164?" reveals the following:

- **Ad Campaign Optimization** (if using Google Ads): If you are using "Maximize Clicks" bidding strategies in Google Ads, the system might be disproportionately sending traffic to this specific zipcode because Google's data indicates a high reach there. The algorithm may prioritize this location to efficiently spend your budget on clicks, even if those clicks don't convert well.
- **Automated Traffic**: The clicks could also originate from bots, monitoring services, or automated systems operating within the data centers located in that specific area. This assumption is compatible with using Bright Data to generate fake traffic. Note that Bright Data is a much better and professional alternative to Tor, whether you use it for good or bad reasons.

1.3.3.3 Other findings

A quick visual inspection of the 2022 dataset immediately revealed an interesting case. It consists of Zscaler, referred to not as a traditional VPN, but as a **modern cloud security platform that can replace VPNs**.

It offers a more advanced and secure way to connect users to both internal applications (using Zscaler Private Access or ZPA) and the public internet (using Zscaler Internet Access or ZIA). ZPA provides a zero-trust, per-application access model that doesn't grant a user access to the entire network, unlike a VPN. The peculiarity of this traffic is as follows, making it a potential candidate to hide questionable activity:

- Several hundred clicks spread across very few zipcodes, with massive clusters of IP addresses specific to each zipcode. For almost all the zipcodes involved, the share of the traffic originating from Zscaler ranges from 60% to 100%.
- Most of the time, the operating systems shows up as Window10, and the browser as Chrome. The zipcodes showing up may be linked to data centers, not to the actual location of the user. They are attached to a few large cities: Atlanta, Chicago, New York, Dallas, Denver, Los Angeles and so on.

I haven't checked yet if this type of scenario, possibly involving similar VPN-like connections from different providers, is widespread in our click reports.

1.3.4 Addendum

In addition to the source code used in my analysis and listed in section 6.3.2, I also summarize the main findings in section 1.3.4.1, and provide a short bio featuring some of my experience in the context of click fraud detection

1.3.4.1 Conclusion

Following a preliminary analysis of about 250k clicks with detailed, high quality data spanning over two time periods, with high granularity and multiple data points (ISP, keywords, device ID, user agent, IP address, campaign ID, time stamps, geolocation, and so on), I came to the following conclusions:

- Many random spikes in traffic thoughout the whole time periods, lasting for several minutes but with a velocity 10 times above average, point to deliberate attacks with a purpose other than defrauding Google, but instead targeted at the advertiser (the plaintiff). These attacks are typical of fraud perpetrated by a competitor. Also, the spikes tend to occur at the same hours in the evening.
- The proportion of devices with multiple IPs is especially large in 2019. However fraud patterns were more subtle in 2022, signaling a shift in the tactics used by the perpetrator: reducing the number of IPs per device, and increased use of VPNs and data centers to hide the source of the traffic. In particular, three zipcodes account for 25% of the traffic in 2022, none of them being known as hubs for golfers. At the same time, some of the most populous US zipcodes (140k inhabitants) show no clicks at all despite expected numbers in the dozens given the total number of clicks.

Next steps may involve looking at CTR, conversion rate and CPC to further confirm the findings and accurately quantify the amount and type of fraud, both in terms of clicks and lost money. Finally, the damage goes well beyond the money lost in fraudulent clicks. Most importantly, the opportunity cost – not getting leads and sales that instead go to the fraudulent competitor – is even more significant in terms of financial impact.

1.3.4.2 Data and Python code

The Python code below is provided to replicate and double-check all my computations. I used it to produce the results, tables, time series and maps discussed in section 1.3.2. The input data used in my program is available to authorized parties only. The main part of the code is in lines 198–211. The code is available here.

```
1  import dns.resolver
2  import dns.reversename
3  import pandas as pd # also requires: pip install openpyxl for Excel
4  import numpy as np
5  import matplotlib.pyplot as plt
6  import matplotlib.transforms as mtrans
7  import matplotlib.dates as mdates
8  import datetime as dt
9
10
11 #-- util function
12 def update_hash(hash, key, count):
13    if key in hash:
14       hash[key] += count
15    else:
16       hash[key] = count
17    return()
18
```

```python

#--
def map_IP_to_dev(df_dev, n_IPs_thresh):

   # Output: IP_clicks: hash with number of clicks per IP
   # Output: IP_flag1: IPs attached to device ID with multiple IPs
   # Input: n_IPs_thresh: number of IPs per device

   dev_hash = {}
   IP_clicks = {}
   clicks_hash = {}
   for index, row in df_dev.iterrows():
      IP = row['IP']
      dev = row['Device Id']
      clicks = row['Clicks']
      update_hash(IP_clicks, IP, clicks)
      update_hash(clicks_hash, dev, clicks)
      if dev in dev_hash:
         local_hash = dev_hash[dev]
         update_hash(local_hash, IP, clicks)
         dev_hash[dev] = local_hash
      else:
         dev_hash[dev] = { IP:clicks }

   IP_flag1 = {}
   for dev in dev_hash:
      n_IPs = len(dev_hash[dev])
      # n_clicks = clicks_hash[dev]
      if n_IPs > n_IPs_thresh:
         for IP in dev_hash[dev]:
            update_hash(IP_flag1, IP, 1) # flag the IP in question
   return(IP_flag1, IP_clicks)


#--
def map_zipcodes(df, IP_flag1, IP_clicks, ratio_thresh, zip_thresh, zip_list):

   # Input: IP_flag1: IPs attached to device with multiple IPs
   # Input: IP_clicks: clicks per IP
   # Input: ratio_thresh: proportion of clicks from a flagged IP, in a zipcode

   import pandas as pd
   df['Time'] = pd.to_datetime(df['Date & Time'])
   zipcodes = []
   colors = []
   dot_sizes = []
   zip_hash = {} # number of clicks per zip (key = zipcode)
   flagged_zips = {}
   for index, row in df.iterrows():
      zip = row['Zipcode']
      IP = row['IP']
      clicks = IP_clicks[IP]
      if IP in IP_flag1:
         update_hash(flagged_zips, zip, clicks)
      update_hash(zip_hash, zip, 1)

   #--- gather data to produce zipcode map

   print("\nTop zipcodes, clicks:\n")
   for zip in zip_hash:
      if zip_hash[zip] > 5:
         click_count = zip_hash[zip]
         # if click_count > zip_thresh:
         if zip in zip_list:
            print(zip, click_count)
         zipcodes.append(zip) # exclude non-US zipcodes?
         if zip in flagged_zips:
            ratio = flagged_zips[zip]/click_count
            if ratio > ratio_thresh:
               colors.append([1, 0, 0])
            else:
               colors.append([1, 1, 0])
         else:
            colors.append([0, 1, 0])
         dot_sizes.append(click_count**0.75)

   #--- produce the map

   # pip install cartopy pgeocode geopandas shapely
   import pgeocode
   import pandas as pd
   nomi = pgeocode.Nominatim("us") # US ZIPs
   df = pd.DataFrame({"zipcode": zipcodes, "color": colors, "size": dot_sizes})
   loc = nomi.query_postal_code(df["zipcode"].tolist())
   df["lat"] = loc["latitude"].values
   df["lon"] = loc["longitude"].values
   df = df.dropna(subset=["lat", "lon"]) # Drop ZIPs without coordinates

   import cartopy.crs as ccrs
   import cartopy.feature as cfeature

   proj = ccrs.LambertConformal()
   fig = plt.figure(figsize=(6.5, 4.5))
   ax = plt.axes(projection=proj)

   # Add basic US features
   ax.add_feature(cfeature.COASTLINE)
   ax.add_feature(cfeature.BORDERS)
   ax.add_feature(cfeature.STATES, linewidth=0.5)

   # Limit extent roughly to continental US (lon_min, lon_max, lat_min, lat_max)
   ax.set_extent([-125, -66.5, 24, 50], crs=ccrs.PlateCarree())

   # Scatter ZIPs as colored dots
   for index, row in df.iterrows():
      ax.scatter(row["lon"], row["lat"],color=row["color"],s= row["size"],
              transform=ccrs.PlateCarree(), # because lat/lon are in PlateCarree
              edgecolor="black",linewidth=0.5,zorder=5,)
   plt.title("Selected US ZIP Codes")
   plt.tight_layout()
   plt.show()
   return()


#--
def plot_time_series(df, threshold):

   # import pandas as pd
   df['Time'] = pd.to_datetime(df['Date & Time'])
   by_hour = df.groupby(pd.Grouper(key='Time', freq='h')).size() # time series data
   by_hour = by_hour.reset_index() # dataframe format
   arr_by_hour = by_hour.to_numpy() # numpy format

   hours = arr_by_hour[:,0]
   clicks = arr_by_hour[:,1]
   colors = []
   print("\nHourly spikes\n")
   for k in range(len(clicks)):
      if clicks[k] > threshold:
         colors.append([1,0,0])
         print(hours[k], clicks[k])
      else:
         colors.append([0,0,0.98])

   for i in range(1,len(hours)):
       plt.plot(hours[i-1:i+1], clicks[i-1:i+1], color='gray', linestyle='-', linewidth=0.1)
       plt.plot(hours[i], clicks[i], marker='o', color=colors[i], markersize=0.4, linewidth = 0.1)

   ax = plt.gca()
   plt.ylabel('Hourly Clicks')
   plt.gcf().autofmt_xdate() # Automatically format date labels
   date_format = mdates.DateFormatter("%Y-%m")
   plt.gca().xaxis.set_major_formatter(date_format)
   current_ticks = plt.xticks()[0]
   new_ticks = current_ticks[::2]
   plt.xticks(new_ticks)
   plt.xticks(rotation=0, fontsize=8)
   plt.yticks(fontsize=8)
   plt.grid(True, linewidth=0.2)
   plt.axhline(y=0, color='gray', linewidth=0.2)
   trans = mtrans.Affine2D().translate(30, 0)
   for t in ax.get_xticklabels():
        t.set_transform(t.get_transform() + trans)
    plt.show()
    return()


#--- Main ---

def collect_data(year):

    if year == 2019:
        file1 = 'Excel1.xls'
        file2 = 'Excel2.xls'
    elif year == 2022:
        file1 = 'Excel3.xls'
        file2 = 'Excel4.xls'

    df1 = pd.read_excel(file1,sheet_name='Web logs')
    df2 = pd.read_excel(file2,sheet_name='Web logs')
    df12 = pd.concat([df1,df2], axis=0)
    df1_dev = pd.read_excel(file1,sheet_name='IPs by number of clicks')
    df2_dev = pd.read_excel(file2,sheet_name='IPs by number of clicks')
    df12_dev = pd.concat([df1_dev,df2_dev], axis=0)

    return(df12, df12_dev)

#--

year = 2022 # options: 2019 or 2022
n_IPs_thresh = { 2019:10, 2022: 1} # IPs per device
ratio_thresh = { 2019:0.1, 2022:0.01 } # prop. of flagged clicks per zip
spike_thresh = { 2019:60, 2022:180} # hourly clicks
zip_thresh = { 2019:400, 2022:900} # clicks per zipcode
zip_list  = ('02143', '07175', '07921', '08054', '10081', '10116', '15295', '20149',
             '22226', '31139', '33231', '64468', '80259', '85001', '90001', '90189',
             '97501', '98004', '98073', '98164', '77494')

(df, df_dev) = collect_data(year)

IP_flag1, IP_clicks = map_IP_to_dev(df_dev, n_IPs_thresh[year])
map_zipcodes(df, IP_flag1, IP_clicks, ratio_thresh[year], zip_thresh[year], zip_list)
plot_time_series(df, spike_thresh[year])
```

1.4 Synthesizing DNA sequences with LLM techniques

This project is not focused on genome data alone. The purpose is to design a generic solution that may also work in other contexts, such as synthesizing molecules. The problem involves dealing with a large amount of "text". Indeed, the sequences discussed here consist of letter arrangements, from an alphabet that has 5 symbols: A, C, G, T and N. The first four symbols stand for the types of bases found in a DNA molecule: adenine (A), cytosine (C), guanine (G), and thymine (T). The last one (N) represents missing data. No prior knowledge of genome sequencing is required.

The data consists of DNA sub-sequences from a number of individuals, and categorized according to the type of genetic patterns found in each sequence. Here I combined the sequences together. The goal is to synthesize realistic DNA sequences, evaluate the quality of the synthetizations, and compare the results with random sequences. The idea is to look at a DNA string S_1 consisting of n_1 consecutive symbols, to identify potential candidates for the next string S_2 consisting of n_2 symbols. Then, assign a probability to each string S_2 conditionally on S_1, use these transition probabilities to sample S_2 given S_1, then move to the right by n_2 symbols, do it again, and so on. Eventually you build a synthetic sequence of arbitrary length. There is some analogy to Markov chains. Here, n_1 and n_2 are fixed, but arbitrary.

1.4.1 Project and solution

Let's look at 3 different DNA sequences. The first one is from a real human being. The second one is synthetic, replicating some of the patterns found in real data. The third one is purely random, with each letter independently distributed from each other, with the same 25% marginal frequency. Can you tell the differences just by looking at the 3 sequences in Table 1.6? If not, see Figure 1.12 and accompanying description.