Bot Hunter Project

Here is some quick context for the reviewer of the assignment that i was unable to cover in the questions below :

Definition of bot click

Since there is no standard definition of bot activity, I have defined bots using the following criteria for ML modeling purposes (*queries for each definition are present in appendix*):

1. Fast Clicks*[1]: Clicks with a time to click of less than 1500 milliseconds are considered bot clicks. This threshold, while arbitrary, is based on the assumption that a human user cannot physically click on an ad that quickly and such behavior is more indicative of automated systems.

2. Bursty Behavior*[2]: Clicks are also classified as bots if there is a burst of activity involving the same search term and ad, defined as 5 or more clicks occurring within 300 seconds or less between each click. Such repetitive, rapid interactions are characteristic of non-human behavior.

Based on the above thresholds, approximately 0.7% (around 1,130 clicks) were classified as bot clicks. These thresholds are based on assumptions. If I had access to a subject matter expert or a verified sample of bot activity, the criteria could be refined further. However, for the purpose of this assignment, these definitions will serve as the working basis for identifying bot clicks.

Note - For a detailed review of the code and logic, please refer to my Jupyter Notebook. It contains the complete analysis, including data cleaning, exploration, target variable definition (i.e., bot clicks), and model building.

1. **Develop a classifier to assess the data for anomalies.**

Here is a quick summary of the ML model in the notebook -

Component	Description	Notes
Objective	Predict the probability that an ad click is generated by a bot.	
Model Type	Random Forest	We can try many more but kept things simple for the initial model
Data Split	70/30 train-test split (stratified)	
Core Features	Search term, clicked ad, server location, browser, device, country, safe search	There were several additional URL parameters such as r, n, ad_exchange, f, bkl, and lspexp1. Since their meanings were unclear, many of them contained null values, and I could not find their

		definitions in the DuckDuckGo parameter documentation, I decided to exclude these features from the model. *Note: I excluded **time to click** from the model because it wouldn't be available in real time. Moreover, since this feature is already used to define bot clicks, including it would introduce data leakage into the ML model.*
Engineered Features	Hour of day, search term length, number of words in search term, presence of numbers in search term, frequency of search term	
Encoding Methods	TF-IDF: search term, clicked ad One-hot encoding: server location, slot ID, browser, device, country, safe search, hour of day	Used TF-IDF since query terms were modified through word- or category-level substitutions, making frequency-based metrics the most suitable for capturing meaningful patterns.
Evaluation Metrics	ROC-AUC: 0.71 Precision: 0.13 Recall: 0.54 F1 Score: 0.21	The model achieves an ROC-AUC of 0.71, indicating it has a moderate ability to distinguish between bot and non-bot clicks. However, the precision of 0.13 suggests that a significant portion of the predicted bots are actually false positives. On the other hand, the recall of 0.54 shows that the model captures more than half of the true bots, which is reasonable for an imbalanced dataset. The F1 score of 0.21 reflects the trade-off between low precision and moderate recall. Overall, the model demonstrates useful discriminatory power but could benefit from further tuning to improve precision without severely impacting recall.

2. **Provide an explanation of the anomalies found, data to back this up, and potential options to filter the anomalous traffic on similar datasets.**

This is quite an open ended question. I am unsure what exactly we mean by "anomaly" here. As per definition it means *something that deviates from normal*.
I spent a significant time cleaning and exploring the data. Here are some things that I felt deviated from normal :

I. To parse the url and the proper field names- I used params from the duckduckgo documentation - https://duckduckgo.com/duckduckgo-help-pages/settings/params . However, not all URL parameters were documented—for example, r, bkl, lspexp1,

atb, and atbva lacked clear definitions. Since their meanings were unclear, I excluded these variables from both the analysis and the machine learning model.

II. Looking at the browser column in data - none of the traffic is from Duckduckgo browser. I am not sure if thats expected or not but I found that a bit strange.

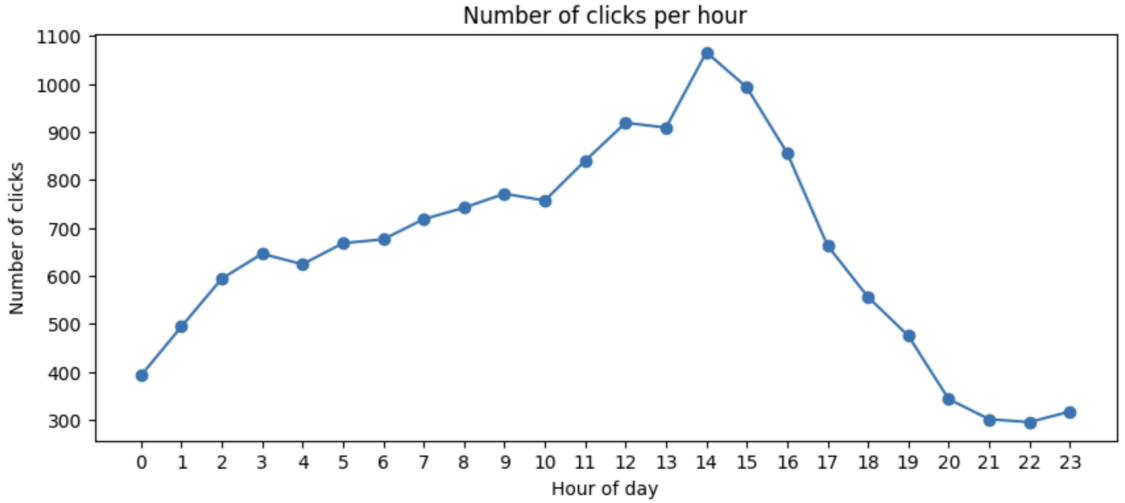
III. Approximately 80% of the clicks have the region labeled as wt-wt (no region). This might be a standard characteristic of DuckDuckGo data, but it appears anomalous. Interestingly, the country column is populated, so it might be possible to infer the region from the country information, although this is uncertain.

IV. TTC parameter in url (or time to click) has a very interesting distribution with minimum ttc as 52 ms and max being 1.5 * 10^9 ms. It seems like the really quick clicks are bots and the long tails seem like some network issue.

```
data["time_to_click"].quantile([i/10 for i in range(0,11)])
```

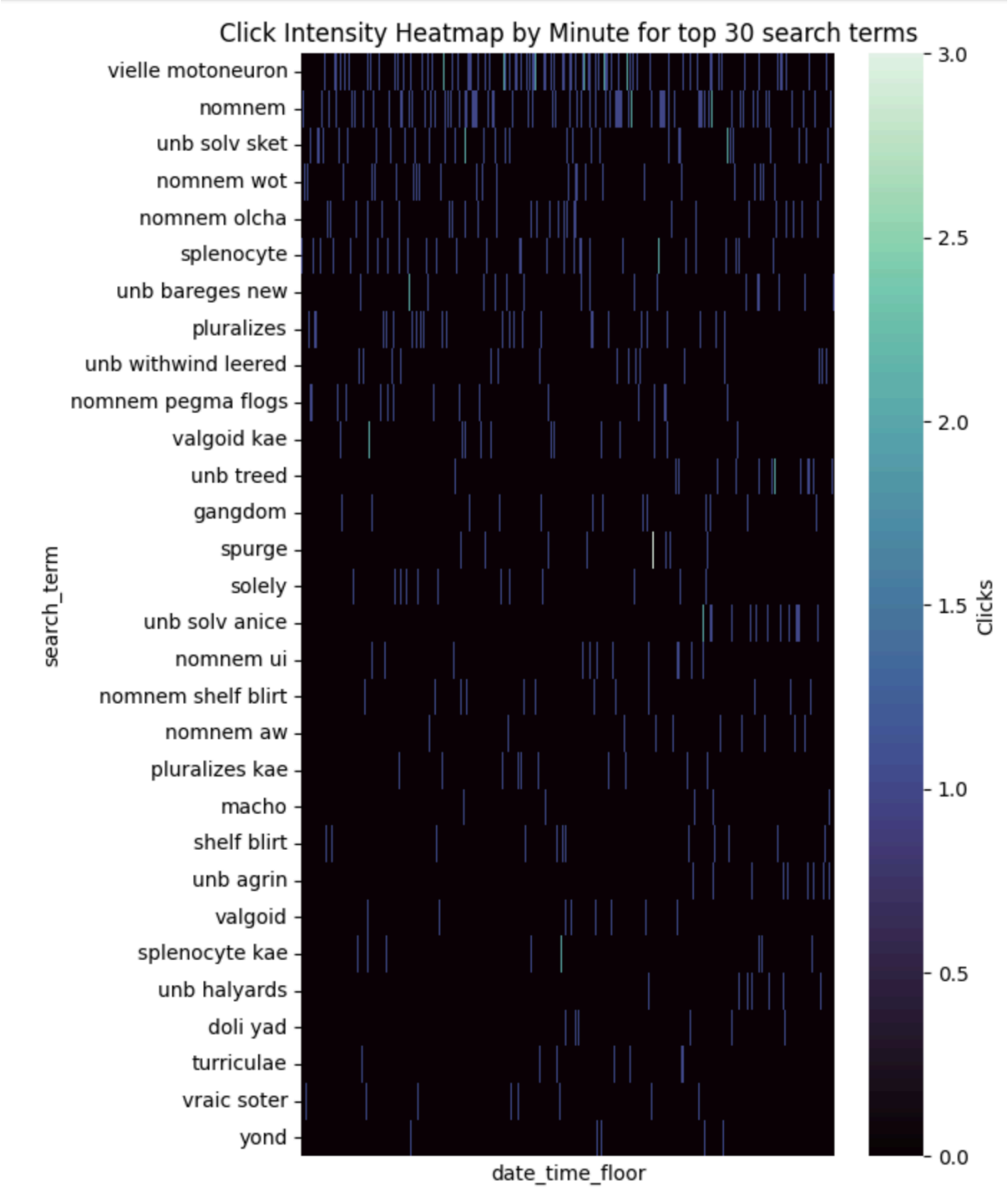
```
0.0           52.0
0.1         1786.0
0.2         2627.0
0.3         3627.2
0.4         5011.6
0.5         7056.0
0.6        11010.4
0.7        15265.0
0.8        15938.2
0.9        24958.4
1.0    154409906.0
```

V. We are not given a timezone of the date time column. *Assuming that it's in the local timezone* we see an interesting trend for clicks when we look at the hours - there is an uptick in activity during the odd hours of the day (1:00 am - 3:00 am).



VI. When analyzing clicks by search term, we observe noticeable bursts in activity for certain terms, which could indicate periods of increased bot activity. In the visualization below, the x-axis represents each minute of the day, while the y-axis

shows the number of clicks per minute, segmented by the top 30 search terms. Thicker bars correspond to higher activity levels within that time frame.



In terms of filtering the data for anomalous traffic we can

I. We can filter out clicks where time to click is exceptionally fast. I have chosen 1.5 s as this threshold. (Insight #4 , also SQL is present in)

II. We can filter out bursts of clicks that seem suspicious as well. I have chosen the threshold as 5 continuous clicks and the time difference between each click is less than 300 seconds.

Honestly, this definition can be made much tighter. But given we had only 15k clicks - if I made the definitions any tighter - we would not have enough bot clicks to do any modelling.

3. **Provide some sort of visualization(s) or visualization tool that would allow a business user (i.e. not a data scientist or engineer) to easily understand your approach and results.**

To avoid repetition here - we can use the viz in question above. Particularly the ones around clicks per hour, quantile distribution of time to click and the heatmap of click activity for top 30 search terms.

4. **Explain how well you think your solution will generalize to other datasets and any shortcomings associated with your approach.**

I believe my approach to defining bot clicks based on fast and bursty click patterns should generalize well to other datasets. The ML model relies on generic features that are typically available across most click datasets. Given the relatively small number of clicks in my data, the threshold I used for defining bot activity isn't very strict. With a larger dataset, I would tighten this threshold to improve the precision and overall quality of the ML model.

5. **Give an assessment of your approach and results from a probability perspective. How likely do you think it is that an individual event your approach flags is actually fraudulent? On what do you base that?**

Based on our definition of bot clicks 0.7% of total clicks. If we use the default cutoff of 0.5 - these are the results from classification report in sklearn

Classification Report:

	precision	recall	f1-score	support
0.0	0.95	0.71	0.81	4344
1.0	0.13	0.54	0.21	339

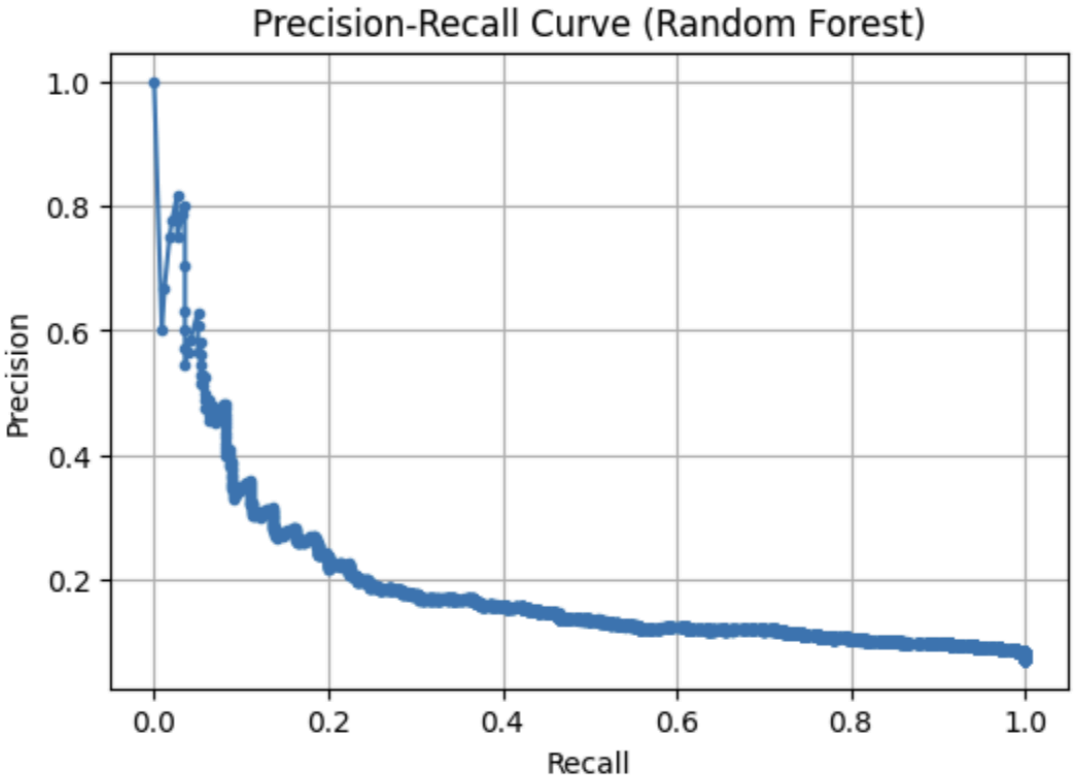
Our precision is 0.13 and recall is 0.54. Therefore **there is a 13% chance that the predicted event is actually fraud**. In other words, when your model flags an event as fraud:

- 13% chance it's actually fraud (true positive)
- 87% chance it's not fraud (false positive)

Given the rarity of fraud (0.7%), our model does reasonably well — it increases the likelihood of fraud from 0.7% (random event) to **13%** (if flagged). That's almost a **19× improvement** in fraud likelihood versus random chance.

Given the very low fraud rate (0.7%), it's inherently difficult to achieve both high recall and high precision. Our current model identifies approximately 54% of all fraudulent clicks, which is a reasonable performance level. However, if false positives are a concern for the business, we can make the model more conservative by increasing the decision threshold. This adjustment would significantly improve precision but reduce recall. For instance, raising the threshold from 0.5 to 0.75 increases precision to around 0.8, meaning **that flagged**

clicks are 80% likely to be fraudulent, but recall drops to 2.6%—capturing far fewer fraud cases than the original 54%. Here is the PR curve to see various PR points:



6. **Given the probability information from step #5, what are some options and your recommendation for how we use the classifier to take actions on bots. If your recommendation relies on business assumptions, please state those assumptions.**

Scenarios Where Launching the Model Makes Sense

1. When the model is used as a triage or prioritization tool
The model increases the likelihood of identifying fraud from 0.7% to 13%, representing a 19× lift. This is highly valuable in workflows where analysts manually review flagged events or apply secondary validation logic. The model effectively narrows the search space and helps focus attention on the most suspicious cases.

2. When the business can tolerate false positives
With a false positive rate of 2.55%, most flagged events will not be fraudulent. However, the model still captures 54% of all fraud, which is strong given the low base rate. If the operational cost of reviewing false positives is acceptable, launching the model is justified.

Scenarios Where the Model Should Not Be Launched

1. When flagged events trigger automatic actions

If predictions lead directly to actions such as auto-blocking, suspending, or denying users, the model's precision of 0.13 is insufficient. In this case, 87% of flagged users would be incorrectly penalized, making the model unsuitable for automated decision-making.

2. When high precision is a business requirement
If the priority is to avoid customer friction or minimize manual review workload, the current threshold is too permissive. While increasing the threshold to 0.75 would raise precision to approximately 0.8, recall would fall to 2.6%, severely limiting the model's ability to detect fraud. (See PR curve above for trade-offs across thresholds.)

7. **Include future work you would like to do given more time and resources.**

Looking ahead, there are several areas I would like to explore:

1. Differentiate between harmful and benign bots: Not all bot activity is detrimental. For example, search-engine crawlers may legitimately scan pages. It would be valuable to identify such benign bots in the dataset and ensure that the model does not take action on them.

2. Refine bot-click definitions with subject matter experts: Collaborating with domain experts would help tighten and refine our current definition of bot clicks and uncover additional patterns or scenarios we may be missing.

3. Incorporate richer click-level signals: Obtaining additional data—such as IP information, historical click behavior, or device fingerprints—would likely improve both recall and precision by giving the model more context.

4. Experiment with more advanced modeling techniques: Exploring more sophisticated ML methods (e.g., ensemble methods, anomaly detection algorithms, sequence models) could further enhance performance, especially for rare-event detection.

Appendix

1. Fast clicks

```
select clicked_ad, search_term, date_time, time_to_click
from data
where time_to_click < 1500
```

2. Bursty Behavior

Logic - we are essentially identifying streaks of 5+ clicks on search term and clicked ad pairs that are less than 300s apart

```
 WITH ordered AS (
          SELECT
            search_term,
            clicked_ad,
            date_time,
            time_to_click,
            ROW_NUMBER() OVER (PARTITION BY search_term, clicked_ad
ORDER BY date_time) AS rn,
            LAG(date_time) OVER (PARTITION BY search_term, clicked_ad
ORDER BY date_time) AS time_lag
          FROM data
),
tagged AS (
          SELECT
            search_term,
            clicked_ad,
            date_time,
            time_lag,
            time_to_click,
            rn,
            date_diff('second', time_lag, date_time) as time_diff,
            SUM(CASE
              WHEN time_lag IS NULL OR date_diff('second', time_lag,
date_time) <= 300
                THEN 1 ELSE 0
            END) OVER (PARTITION BY search_term, clicked_ad ORDER BY
date_time) AS click_count
          FROM ordered
),
grouped AS (
          SELECT
            tagged.*,
            rn - click_count AS grp_id
          FROM tagged
),
bursts AS (
    SELECT
            search_term,
            clicked_ad,
            grp_id,
            count(*) as streak
          FROM grouped
          WHERE time_diff <= 300
          GROUP BY search_term,
            clicked_ad,
            grp_id
```

```
            HAVING count(*) >= 5
)

SELECT g.search_term, g.clicked_ad, g.date_time, g.time_to_click
FROM grouped g
JOIN bursts b
  ON g.search_term = b.search_term
  AND g.clicked_ad = b.clicked_ad
  AND g.grp_id = b.grp_id
ORDER BY g.search_term, g.clicked_ad, g.date_time;
```