Non intrusive Click-Bots detection

Amine Boujida
a.boujida.f@gmail.com

Ad click Bots are everywhere in the internet with over 50 percent of the traffic classified as bot , In addition, 67% of bad bot traffic originates from public data centers in North America. [1][2]. Most of the techniques for bot detection rely on intrusive data-collection (mouse/keyboard hooks, CAPTCHA challenges, client-side fingerprinting), which makes the task in a privacy-preserving set-up challenging[3].
We have a one-day sample of 15 000 clicks, sourced from a publisher, with anonymized attributes (Query, Region etc..)

```
{
    sample 1: 2019-12-02 13:04:19, Mars, Safari, Android,
          /ad_click?n=1&f=1&d=www.heathrowexpress.com&...&ttc=2983&q=trickily%20voetian&...
    sample 2: 2019-12-02 19:41:24, Mercury, Chrome, Android,
          /ad_click?n=1&f=1&d=octopart.com&...&ttc=18033&q=fo&...
}
```

The goal is to identify bot clicks and uncover their underlying patterns. In the presence of ground truth, we could treat this as a supervised classification problem—aside from the usual label-imbalance issues, most of the guidance we would need would figure in the positive samples.

In the absence of labels, we can still apply unsupervised methods such as k-means to detect anomalous clusters. However, bot clicks may not form compact, well-separated groups in high-dimensional space, and at this stage we need insights first, using as little assumptions as possible, therefore, simple clustering techniques might not be ideal for that.

It becomes clear that domain priors —what types of bots we expect, where they come from, and how they behave—are essential to guide both feature design and our choice of anomaly-scoring method.

Let's think about why bots would ever visit a search engine and click on ads, in order to get an idea of how they would manifest in our feature space: While some benign crawlers discover search engines as entry points (these are less likely to click on ads), most malicious clicks fall into two camps: publishers inflating their own metrics, or competitors attempting to exhaust an advertiser's budget [4, 5]. In our setting (not a random publisher likely to inflate their metrics), budget-draining (competitor-driven) bots seem far more plausible—they often reuse shared infrastructure such as click farms and botnets [6, 7], hammer a very narrow set of ads (and thus exhibit much lower query diversity than normal users), and produce bursts of clicks tightly clustered in time rather than the broad temporal variability of humans. More sophisticated bots will try to mimic human timing and interaction patterns.

Let's think of how these bots would harm the business—that will help us gauge how sophisticated our detection must be. Most advertisers already perform click-invalidation in-house, but if a particular publisher (or IP range) generates a persistently high invalidation rate, ad networks may penalize or even delist them to protect their reputation and maintain advertiser. Blocking bad clicks before they ever reach the ad network demands highly calibrated, high-precision scoring—otherwise you risk (a) rejecting too many legitimate clicks, or (b) letting fraud slip through. By contrast, if our primary goal is simply to estimate the overall scale of fraudulent activity (rather than to block every bad click), we can afford coarser "risk scores" with lower calibration—useful as a first pass to inform roadmap and resource allocation. Later, we can cross-validate these high-level estimates against known low conversion rates for bots or with security-team risk assessments.

The following work builds on these assumptions: We started by evaluating a simple baseline using prior knowledge about the time-to-click (TTC) distribution. Extremely fast clicks are often an indicator of bot

activity. We also observed that bot clicks tend to concentrate in a very narrow TTC interval, which is inconsistent with natural user behavior. Additionally, we examined query duplication, where certain search strings were repeated much more frequently than expected, suggesting automated "hammering" of specific keywords, aligning with the budget-exhaustion hypothesis. We then fitted a machine learning model that provided deeper insights, completing the image we obtained from the baseline. Section 1 discusses the initial exploratory analysis and the approach used to detect anomalous activity in the ad-click data. Section 2 outlines the baseline models and feature engineering process used to identify suspicious activity. Section 3 presents the machine learning model that enhances fraud detection beyond the baseline. Section 4 provides insights into how the anomalies detected by the model can be interpreted. Section 5 covers the probability calibration of the model. Section 6 addresses the assumptions and limitations of the approach, while Section 7 explores future work under identical constraints.

1 Exploratory analysis

After processing the raw URL strings we recovered over a dozen parameters. We inferred likely meanings based on the values they take, and left a few opaque parameters in the DataFrame in case they prove useful later. Distribution of each feature in the appendix. A.1

Parameter	Inferred Description	Example Values
`n`	unknown	1
`f`	unknown	1
`d`	target domain	`www.adlibris.com`
`sld`	unknown	0,1
`st`	search type / source	`mobile_search_intl`
`nt`	unknown	0
`r`	unknown	0,1,2
`adx`	ad exchange	`none`, `def`, `oag`
`adx_name`	ad exchange name	`none`, `adsb`
`ttc`	time-to-click (ms)	1228, 7297, 1599
`q`	user query	`spotsmen`, `jower indigen`
`ct`	country code	`SE`, `ES`, `NL`
`kl`	region	`se-sv`, `wt-wt`
`kp`	safe-search flag	-1, 1
`lsexp1`	experiment flag	(empty), `b`, `a`
`bkl`	inferred : position of the ad	(empty), `r1-0`, `ra-0`
`om`	unknown	(empty), `news:i:1`
`atb`	unknown	(empty), `REDACTED`
`va`	unknown	`a`, `b`, `c`
`atbva`	unknown	`k`, `p`, `g`
`atbexp`	unknown	`uiu`

Table 1: Ad-click URL parameters: inferred meanings and example values

We divide our extracted parameters into two sets:

- **Strong signals:** These features capture direct evidence of automation. For example:
  - *Time-to-click (TTC):* simple bots will often click within a few hundred milliseconds of page load, whereas human TTC is almost never below 500.
  - *Query duplication count* (`q_count`): repeated identical queries in rapid succession are a tell-tale sign of scripting.
  - *Domain duplication count* (`d_count`): high rates of revisiting the same landing domain that is otherwise unpopular

  - *Click patterns* (`d_count`): high number of clicks in a short amount of time coming from a concentrated intersection of spatial features(region , device , browser , domain ...)

- **Covariates:** Contextual attributes—region code, browser, device type, hour-of-day, etc.—help us validate that our flagged positives form a coherent subgroup. Consistency (e.g. a narrow region distribution, uniform browser profile, synchronized diurnal pattern) in the positive class, compared to the overall population, reinforces our automated-behavior hypothesis.

Key assumptions

1. **Machine-speed automation.** Bot scripts generate clicks with millisecond-level timing and extremely regular inter-click intervals, unlike human users whose reaction times and pause patterns vary widely.

2. **Shared infrastructure.** Competitor-driven and publisher-driven bots often run on common IP ranges (botnets or click farms) and thus exhibit concentrated spikes in region/browser/device space.

Further details about Assumptions and limitations are discussed in 6. We establish the following metrics to validate our core assumptions:

Metrics

1. **Normalized entropy.** Measures how concentrated a distribution is over m categories. If all mass sits in a single bin, it is 0; if it is perfectly uniform, it is 1. In practice, for a discrete probability vector $P = (p_1, \ldots, p_m)$ we compute

$$H_{\text{norm}}(P) \;=\; \frac{-\sum_{i=1}^{m} p_i \ln p_i}{\ln m}\,.$$

   Low values of H_{norm} on our "positive" (i.e. bot-like) subset confirm that those samples are tightly clustered along that feature.

2. **Total variation distance.** Captures the overall mass discrepancy between two histograms—i.e. the largest share of clicks that would need to be "reassigned" to turn one distribution into the other. Given two empirical probability mass functions $P = (P_1, \ldots, P_m)$ and $Q = (Q_1, \ldots, Q_m)$ over the same categories (e.g. region codes), we define

$$\text{TV}(P, Q) \;=\; \tfrac{1}{2} \sum_{i=1}^{m} \big|P_i - Q_i\big|.$$

   For example, if 30% of bot clicks come from a particular region but only 10% of normal clicks do, that 20% mismatch contributes directly to TV. A high TV on `region` implies our "bot" vs. "non-bot" labels align strongly with geographic clustering—an actionable signal that certain IP blocks or data-center locales are over-represented in fraud. We average TV over all chosen covariates to yield a single separation score that quantifies how distinct our anomaly set is from the baseline traffic.

2 Baseline model

During exploratory analysis two interesting patterns emerged:

- **High query repetition.** Certain search strings occur hundreds of times—far more than a typical long-tail distribution would predict—suggesting automated "hammering" of specific keywords.

- **Discontinuous TTC spikes.** The time-to-click (TTC) histogram exhibits sharp, isolated peaks rather than a smooth decay, which is inconsistent with natural human response times and points to scripted or coordinated click bursts.

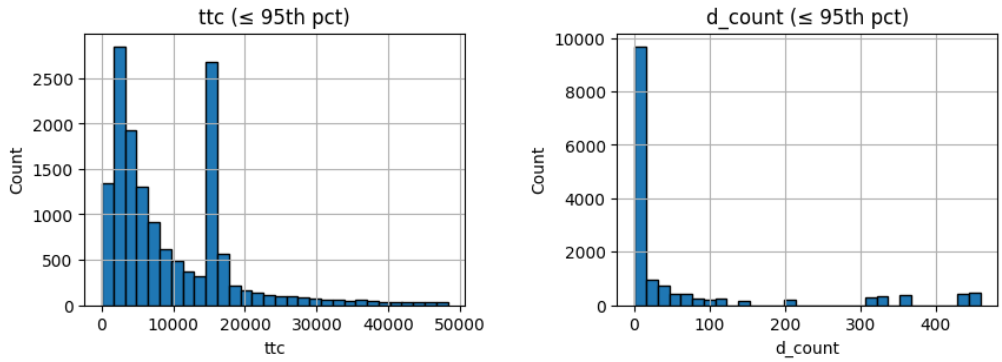


Figure 1: Distributions : ttc and query duplication count

For a baseline model we use the following signals:

Feature engineering

- **Time-to-click (TTC).** First, a TTC under 500 ms is most likely a bot click. Human reaction times naturally follow a broad, log-normal curve [8], so genuine users' clicks scatter over a wide range of delays. By contrast, automated clickers tend to fire with almost identical TTCs—far from the human mean and with unnaturally low variance. As our first baseline signal, we compute for each click a "TTC-anomaly" score (see A.2 for details).
- **Query duplication.** At first glance, seeing the exact same query—especially a longer, multi-word phrase—repeated many times seems unlikely. To quantify this intuition, we use length-stratified repeat-rate curves from the AOL query log [10] to assign each click a "duplication-anomaly" score A.3.

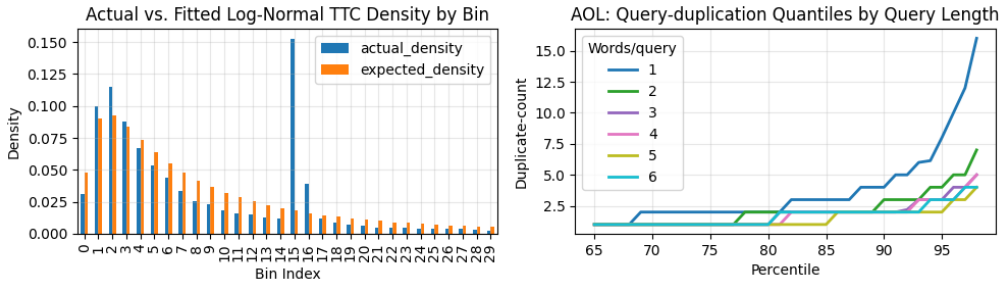


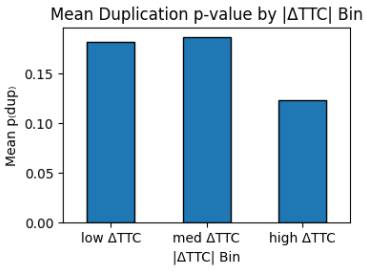
Figure 2: Left: fitted log-normal TTC model. Right: duplication-count quantiles by query length.

We observe a clear spike in TTC around 15000ms, which the log-normal error signal captures well. Likewise, the duplication quantiles show that very short queries (1–2 words) repeat far more often, while longer queries become increasingly unlikely to be duplicated.

Note1: We assume that the anonymization process preserves each query's original word count, so the duplication signal remains valid.

```
{'q': 'pogonias motorcab', 'q_count': 1,'no_words_query': 2,
    'ttc': 6207,'expected_density': 0.0554,'actual_density': 0.0442,
    'delta_density_ttc_bin': -0.0112,'pval_dup_count_query': 0.2277},
  {'q': 'henries','q_count': 1,'no_words_query': 1,
  'ttc': 306,'expected_density': 0.0477,'actual_density': 0.0314,
    'delta_density_ttc_bin': -0.0164,'pval_dup_count_query': 0.3168},
  {'q': 'blade m hd','q_count': 2,'no_words_query': 3,
    'ttc': 1062,'expected_density': 0.0905,'actual_density': 0.1002,
    'delta_density_ttc_bin': 0.0097,'pval_dup_count_query': 0.0891},
  {'q': 'condiddle spondaize','q_count': 1,'no_words_query': 2,
    'ttc': 527,'expected_density': 0.0477,'actual_density': 0.0314,
    'delta_density_ttc_bin': -0.0164,'pval_dup_count_query': 0.2277}
```


(a) Example baseline signals



(b) TTC-anomaly vs. duplication p-value

Figure 3: (a) How individual clicks are encoded for our baseline (JSON). (b) Their joint behavior—clicks with high TTC-anomaly also tend to have low duplication p-values.

Results :

We perform a grid-search over the TTC-anomaly threshold τ_δ and the duplication p-value threshold τ_p, choosing the pair that minimizes the combined metric

$$\text{score} \;=\; \tfrac{1}{2}\big(\overline{\text{TV}}\big) \;+\; \tfrac{1}{2}\big(1-\overline{H}\big),$$

where $\overline{\text{TV}}$ is the average total-variation distance across our covariates and $\overline{H}$ is the average normalized entropy of the "bot" group (we wish to minimize entropy).

The optimal cutoffs are

$$\tau_p = 0.010, \quad \tau_\delta = 0.023,$$

which flags 2% of clicks as anomalous (see Appendix B.1 for more details). Accordingly, our baseline classifier is

$$\widehat{\text{bot}}(i) = \begin{cases} 1, & \text{if } TTC_i < 500 \text{ ms} \quad \text{or} \quad \big(p_i < \tau_p \;\wedge\; \Delta_{\text{TTC},i} > \tau_\delta\big), \\ 0, & \text{otherwise.} \end{cases}$$

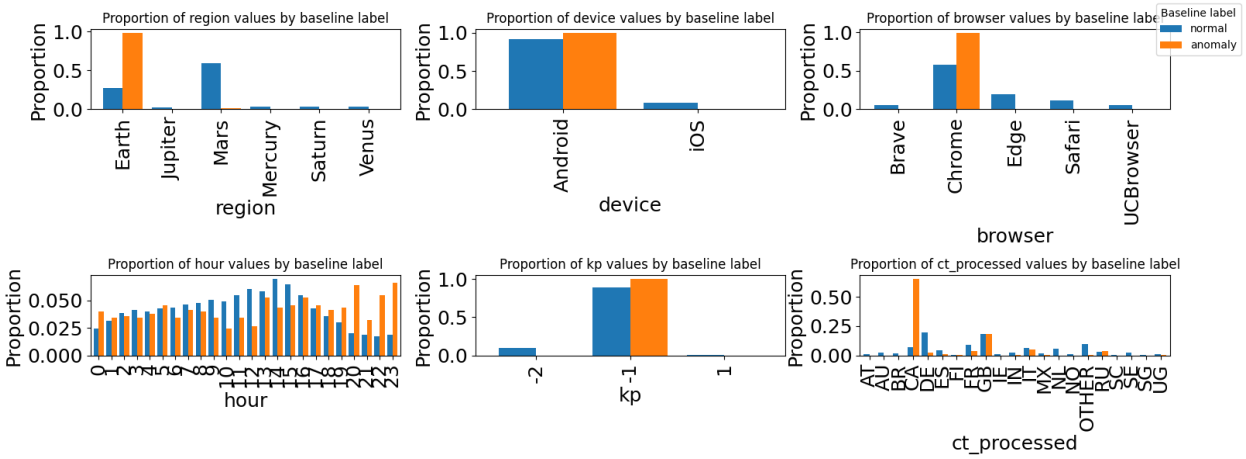


Figure 4: Covariates Distributions comparison See Note 2 for more details.

Across the six covariates shown, the anomaly (positive) class exhibits much tighter, less varied distributions than the "normal" (negative) class:

- **Region:** Anomalous clicks come almost entirely from a single region, while normal traffic is spread across multiple locales.

- **Device & Browser:** Nearly every anomaly originates on Android/Chrome, in contrast to the mixture of Android + iOS and multiple browsers seen in the negative class.
- **Hour of Day:** The anomaly profile is almost flat and shifted toward late-evening hours, whereas normal clicks follow the familiar daytime bell curve.
- **"kp" safe search Flag:** Anomalous sessions are locked at a single `kp = -1` value, but human traffic shows all three levels (`-2`, `-1`, `1`).
- **Country Code (`ct_processed`):** Anomalies cluster tightly in one or two country codes (e.g., "CA"), while normal clicks span many more codes.

Note2: Covariates Distributions comparison : Botty clicks in orange - Orange bars sum to 1 and blue bars sum to 1, we fix the class (positive/negative) and we compute the distribution within each. The plots display normalized distributions per class rather than raw counts, enabling direct comparison of how feature values are distributed differently between normal traffic (blue) and botty/anomalous clicks (orange). This normalization approach highlights which feature values are disproportionately associated with each class, revealing distinctive patterns even when class sizes differ significantly.

The bot behavior is expected to manifest in many other dimensions, we leverage the power of ML in the next section to investigate this .

3 Bot detection using ML

We use `IsolationForest` because it makes minimal assumptions—only that anomalies (few and different) tend to be isolated in fewer random splits—without requiring any density estimate or clustering structure [9]. It assigns each sample i an anomaly score proportional to the average path length across t random trees: shorter paths $\Rightarrow$ more anomalous.

Key parameters [11]:

- `contamination`: fraction of samples expected to be anomalies; sets the decision threshold on the scores.
- `n_estimators`: number of trees in the forest; more trees improve score stability but increase runtime.
- `max_samples`: Controls how many training instances each tree sees. A small value (e.g. a few hundred) makes each tree focus on local structure—quickly isolating extreme outliers but increasing variability between trees—while a large value (up to the full dataset) yields more stable scores by capturing the global distribution, at the risk of diluting rare-event signals.
- `max_features`: Determines the fraction of features considered at each split. Using few features (e.g. 10–30 %) injects more randomness and helps detect anomalies that stand out strongly on individual dimensions, whereas using many features (e.g. 70–100 %) lets splits exploit complex, multi-feature patterns—improving detection of subtle outliers but reducing tree diversity.
- `bootstrap`: sample with replacement when drawing each tree's subset; yields greater diversity among trees.

Feature Encoding: Continuous variables are kept unchanged, while categorical variables are encoded using frequency encoding. After experimenting with dummy encoding for categorical variables, the results were inconclusive, mainly due to the high number of features and the increased complexity of the model, which reduced interpretability. Frequency encoding aligns better with the model's logic, as it performs random splits based on the frequency of feature occurrences.

Metrics :

In addition to the previously defined signals `pval` and `delta`, we track the three following metrics to assess model performance:

Silhouette score: This metric measures how well-separated the anomalies (detected by the model) are from the rest of the data. A higher silhouette score indicates that the anomalies are distinct from the normal data points. [0].

KMeans intersection: We perform KMeans clustering with two clusters and compare the clustering result with the anomaly mask. The *intersection* represents the fraction of anomalies that are also identified as the same cluster by KMeans. This metric helps to quantify how well the model's anomaly detection aligns with the clustering approach, and ideally, a higher intersection suggests consistency between both methods.

Disagreement: This metric calculates the average disagreement between multiple runs of the Isolation-Forest model with different random states. It quantifies how consistent the anomaly detection results are across different initializations. A lower disagreement indicates more stable anomaly detection performance.

Results :

The following set of parameters maximizes the aggregated score (entropy and TV distance):

Feature Set	`pval_dup_count_query`, `delta_density_ttc_bin`, `q_count`
	`d_count`, `encoded_binned_ttc_count`, `encoded_kp_count`
	`encoded_region_count`, `encoded_bkl_count`, `encoded_om_count`
	`hour`, `no_words_query`

Model Parameters	`n_estimators`: 1000, `max_samples`: 1000, `max_features`: 0.1 `bootstrap`: True,

Feature Set	`pval_dup_count_query`, `delta_density_ttc_bin`, `q_count`
	`d_count`, `encoded_binned_ttc_count`, `encoded_kp_count`
	`encoded_region_count`, `encoded_bkl_count`, `encoded_om_count`
	`hour`, `no_words_query`

Evaluation Metrics	`bot_rate`: 0.07, `avg_entropy`: 0.5004, `avg_TV_dist`: 0.1581 `kmeans_intersection`: 0.5311, `silhouette`: 0.5195, `disagreement`: 0.0079 `avg_pval_pos`: 0.0218, `avg_pval_neg`: 0.174, `avg_delta_pos`: 0.0531 `avg_delta_neg`: 0.0186, `agg_score`: 0.3288

Table 2: Optimized Parameters and Metrics

See detailed grid search results C.1

In contrast to the baseline, where we observed a very narrow region of TTC (around 15,000) for the positive class, the IsolationForest model displayed greater flexibility in capturing anomalies across a broader range of values.

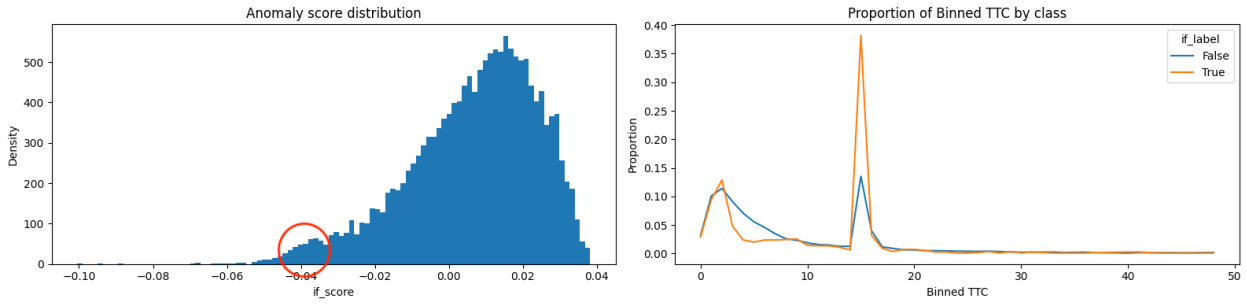


Figure 5: Anomaly score distribution on the left and Binned TTC distribution comparison on the right

The anomaly score distribution shows a sharp peak at the center, with a slight bump in the left (high anomaly) region, indicating a small cluster of highly anomalous points.

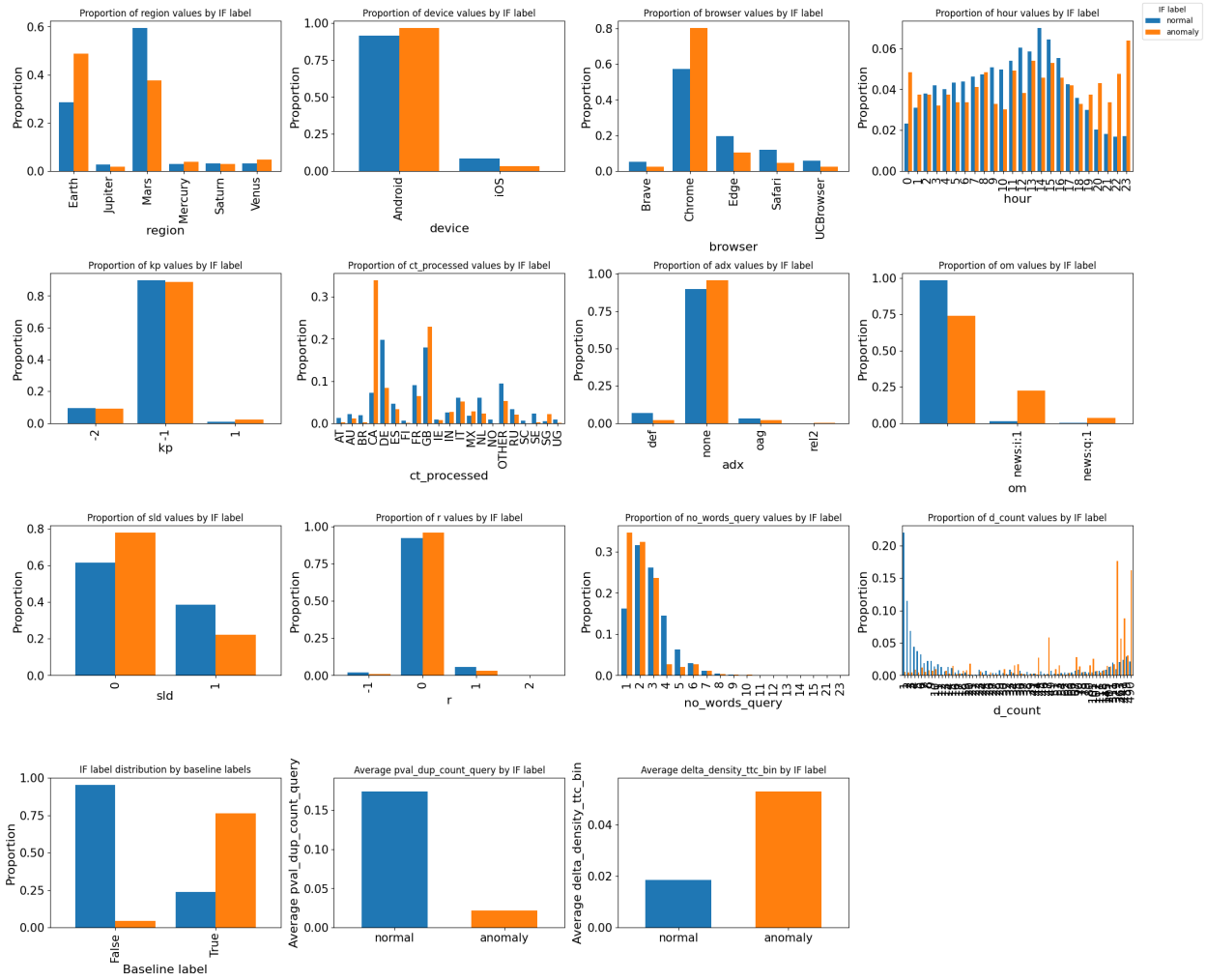


Figure 6: Covariates Distributions comparison See Note 2 for more details.The baseline label plot chart shows that when IsolationForest (IF) predicts “normal” (blue), the baseline rarely identifies a bot—suggesting strong agreement in labeling non-bot clicks. On the other hand, when IF predicts an “anomaly,” it captures most of the baseline’s “bot” labels (orange), along with some additional outliers that the baseline misses. In the Region plot, ”Mars” traffic is primarily associated with normal behavior, while ”Earth” shows a higher proportion of anomalous activity. The hour-of-day distribution highlights that midday hours (11–15) are more likely to represent normal traffic, while the late-night hours (0 and 23) show an increase in anomalies. The Query length distribution suggests that anomalies are more likely to stem from single-word searches, while multi-word queries are more prevalent in normal traffic. The Device and browser distributions indicate that Chrome users exhibit higher anomaly rates compared to Edge users. The OM plot shows a high difference in distributions between the 2 classes. Lastly, the Domain count plot reveals that anomalies tend to be associated with higher domain counts—indicating users interacting with many distinct domains—potentially pointing to a budget-exhaustion pattern. Normal traffic, by contrast, is more concentrated in the lower domain count range.

4 Explainability

In addition to previous insights 3 2 we investigated the 2D umap projection of the feature space :

Embedding space To uncover intrinsic structure in our data independent of the baseline signals or any labels, we applied a 2D UMAP [0] projection to the feature vectors. This low-dimensional embedding

reveals clusters and outlier regions purely driven by the raw covariate relationships. The figure below is a static snapshot of the resulting layout, color-coded by the IsolationForest anomaly score. For an interactive version—complete with per-point metadata and zoom/pan controls—see `embedding_space.html` in the code repository[15].

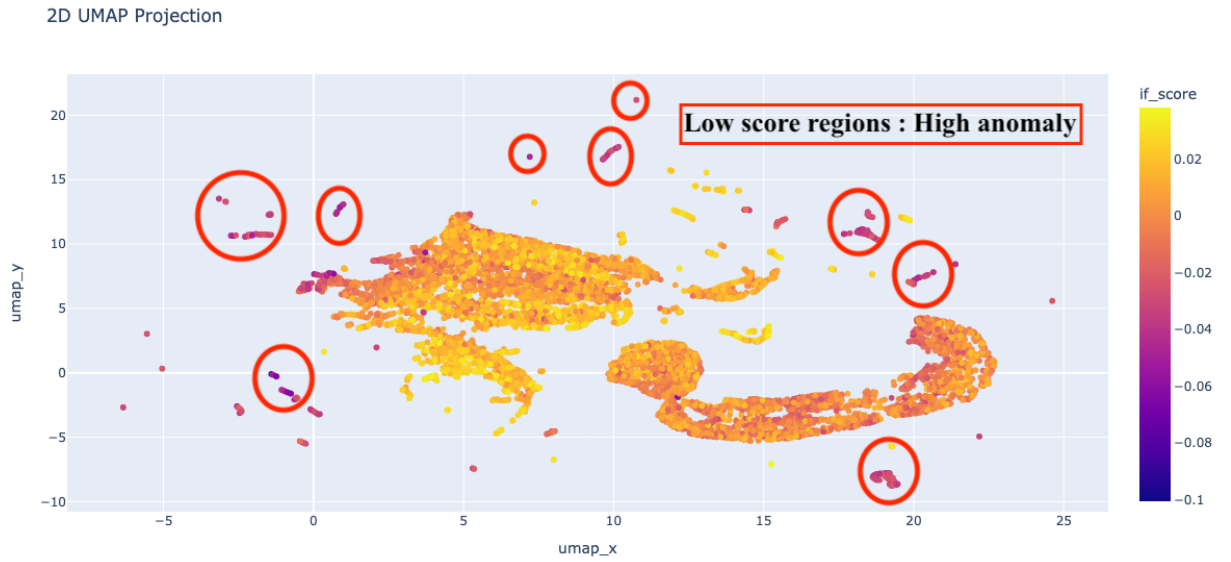
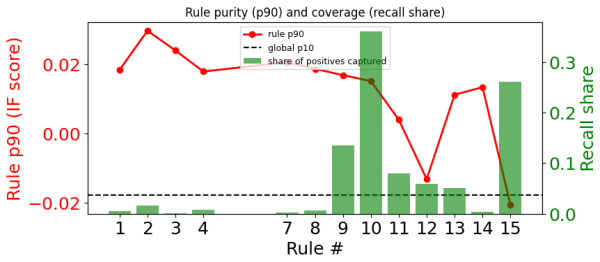


Figure 7: Umap 2D embedding space

Filters In addition to the TTC baseline, we extract simple filters directly from the Isolation-Forest scores. SHAP pinpoints the columns that matter most. With those features we fit a *shallow* decision tree on the raw Isolation-Forest scores and turn each leaf into an *IF ... THEN ...* rule. For every rule we look at the 90$^{\text{th}}$ percentile of its scores: when this $p90$ is very low, ninety percent of the clicks covered by that rule already sit deep in the anomaly zone. Rules with the lowest $p90$ therefore catch many bots while adding few false positives.



Rule #	Definition
1	duplicated_query ≤ 0.500 ∧ region_is_Venus ≤ 0.500 ∧ hour ≤ 19.500 ∧ hour ≤ 0.500
2	duplicated_query ≤ 0.500 ∧ region_is_Venus ≤ 0.500 ∧ hour ≤ 19.500 ∧ hour > 0.500
3	duplicated_query ≤ 0.500 ∧ region_is_Venus ≤ 0.500 ∧ hour > 19.500 ∧ ttc ≤ 7013.500
4	duplicated_query ≤ 0.500 ∧ region_is_Venus ≤ 0.500 ∧ hour > 19.500 ∧ ttc > 7013.500
7	duplicated_query ≤ 0.500 ∧ region_is_Venus > 0.500 ∧ ttc > 368.500 ∧ ttc ≤ 12367.500
8	duplicated_query ≤ 0.500 ∧ region_is_Venus > 0.500 ∧ ttc > 368.500 ∧ ttc > 12367.500
9	duplicated_query > 0.500 ∧ ct_processed_is_CA ≤ 0.500 ∧ delta_density_ttc_bin ≤ 0.023 ∧ browser_is_Chrome ≤ 0.500
10	duplicated_query > 0.500 ∧ ct_processed_is_CA ≤ 0.500 ∧ delta_density_ttc_bin ≤ 0.023 ∧ browser_is_Chrome > 0.500
11	duplicated_query > 0.500 ∧ ct_processed_is_CA ≤ 0.500 ∧ delta_density_ttc_bin > 0.023 ∧ ct_processed_is_GB ≤ 0.500
12	duplicated_query > 0.500 ∧ ct_processed_is_CA ≤ 0.500 ∧ delta_density_ttc_bin > 0.023 ∧ ct_processed_is_GB > 0.500
13	duplicated_query > 0.500 ∧ ct_processed_is_CA > 0.500 ∧ delta_density_ttc_bin ≤ 0.023 ∧ kl_processed_is_ca-en ≤ 0.500
14	duplicated_query > 0.500 ∧ ct_processed_is_CA > 0.500 ∧ delta_density_ttc_bin ≤ 0.023 ∧ kl_processed_is_ca-en > 0.500
15	duplicated_query > 0.500 ∧ ct_processed_is_CA > 0.500 ∧ delta_density_ttc_bin > 0.023 ∧ ttc ≤ 16137.000

Figure 8: **Left:** For every decision-tree rule, the red line shows the 90$^{\text{th}}$-percentile Isolation-Forest score (*lower = purer*); the dashed line is the global 10$^{\text{th}}$-percentile cut used to label bots. Green bars give the fraction of all labelled bots that the rule captures (recall share). **Right:** Exact IF-style filter behind each rule ID. Rules whose red point lies below the dashed line grab mostly anomalous clicks while covering a meaningful share of bot traffic.

This new rule captures the largest share of positives (26 Combined with the original baseline **??**, the merged rule set now catches 37 % of the labeled bots while ensuring a low False positives rates (P90 of score being very low):

$$\widehat{\text{bot}}_{\text{new}}(i) = \begin{cases} 1, & \text{TTC}_i < 500 \text{ ms} \quad \vee \quad \big(p_i < \tau_p \wedge \Delta_{\text{TTC},i} > \tau_\delta\big) \\ & \quad \vee \quad \big(\text{duplicated\_query}_i = 1 \wedge \text{ct}_i = \text{CA} \wedge \Delta\text{density}_{\text{TTC-bin},i} > 0.023 \wedge \text{TTC}_i < 16{,}137\big), \\ 0, & \text{otherwise.} \end{cases}$$

5 Probability calibration

A variety of ad-hoc and principled methods exist to turn raw anomaly scores into interpretable probabilities. The simplest is the percentile-rank, which assigns to each score s_i the fraction of samples with score $\leq s_i$. While trivial to compute and free of distributional assumptions, percentile ranks become unstable in the extreme tail.

Because we believe fraud concentrates in those rare, extreme scores (after inverting $s \mapsto -s$), we cannot rely on a KDE there (it either oversmooths or collapses to zero). Instead, Extreme Value Theory tells us that the excesses above a high threshold u follow a Generalized Pareto distribution (GPD), which provides a smooth, extrapolatable model for the tail density.

In practice we then compute

$$P(\text{fraud} \mid s) = \frac{P(s \mid \text{fraud})\, P(\text{fraud})}{p(s)},$$

setting $P(\text{fraud}) = 0.02$ (business prior) , estimating $p(s)$ by a Gaussian KDE on the full dataset, and approximating $P(s \mid \text{fraud}) \approx p(s \mid s > u)$ via the fitted GPD. This lightweight but rigorous approach is expected to yield better probability estimates that we can rely on for follow up actions.

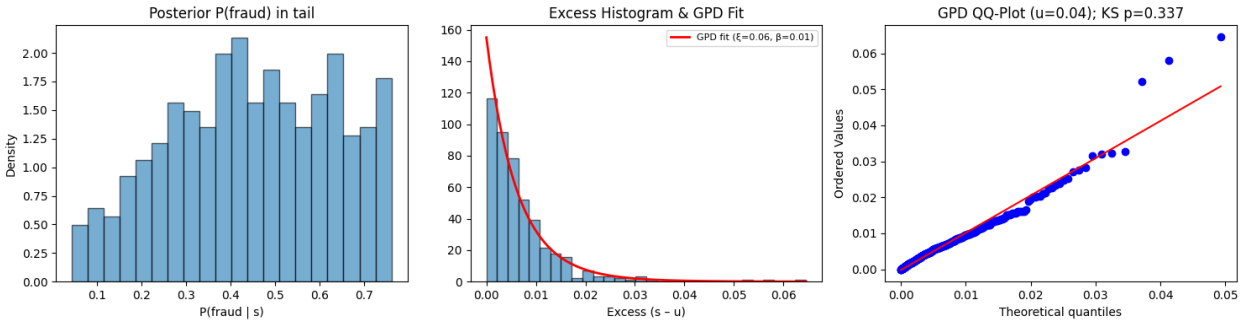


Figure 9: Fraud-probability modeling in the extreme tail. **Left:** Histogram of the posterior $P(\text{fraud} \mid s)$ for all clicks with inverted IF score s above the 97.5th percentile threshold u. **Middle:** Histogram of the excesses $s - u$ (bars) with the fitted Generalized Pareto density (red curve), illustrating how the GPD captures the shape of the extreme tail. **Right:** QQ-plot of empirical excesses versus theoretical GPD quantiles. The near-diagonal alignment and the Kolmogorov–Smirnov test result ($D = 0.047$, $p = 0.337$) confirm a statistically sound fit. We selected u by scanning high quantiles until the KS p-value exceeded 0.05, ensuring our parametric tail model is well justified before using it as the likelihood $p(s \mid \text{fraud})$.

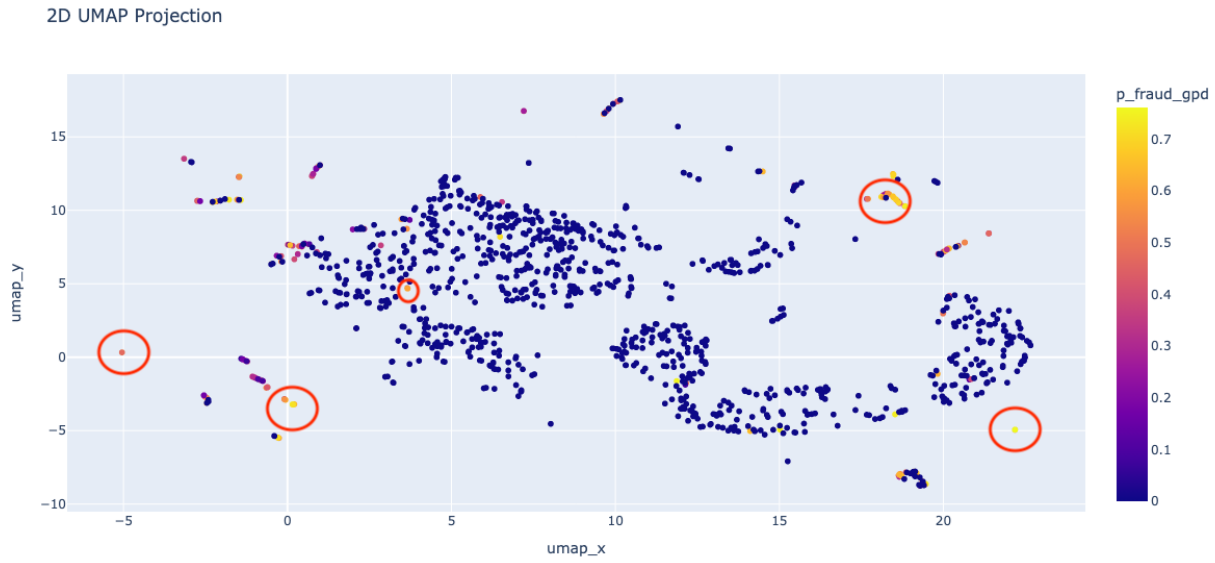


Figure 10: UMAP embedding of click events colored by GPD-based posterior fraud probability. High-probability points (yellow–red) form a few small, isolated clusters, while the vast majority remain blue at zero since the GPD tail model only applies beyond the extreme threshold. Unlike the continuous gradient of raw IsolationForest scores 4, this posterior sharply isolates only the most extreme outliers.

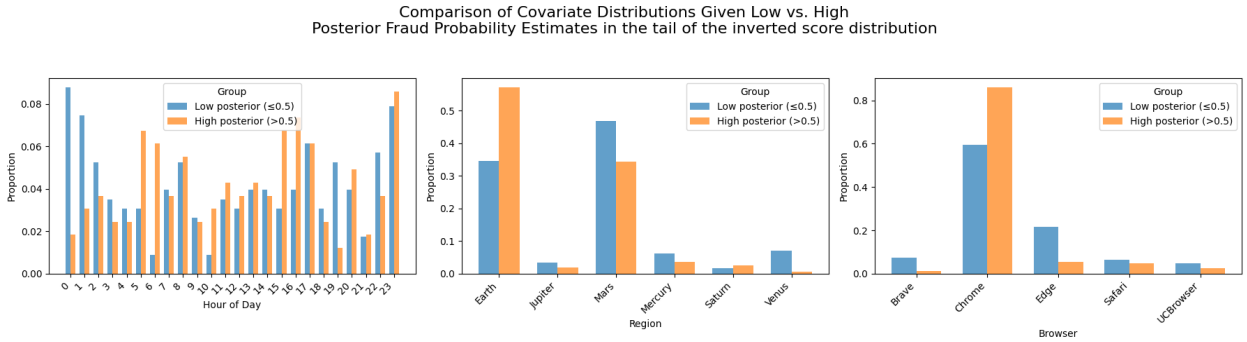


Figure 11: Applying a $p_{\text{fraud\_gpd}} > 0.5$ cutoff reinforces earlier patterns: high-risk clicks cluster in late hours, overwhelmingly come from "Earth," and are predominantly on Chrome.

6 Assumptions and limitations

- The second baseline signal—the p-value based on (word count, query duplication count)A.3—may seem complex at first, but we added it to prevent overfitting to the TTC signal and to counterbalance its obvious bias around 15 000 ms.
- The choice of metrics—total variation distance to highlight group differences and normalized entropy to measure purity—was intentionally simple. These could be extended with diversity metrics (for example bot activity may not concentrate in a single region), or by reporting per-feature TV/entropy rather than a single aggregate; alternative aggregation schemes could better reflect business priorities.
- Our analysis yields data-driven hypotheses about potentially fraudulent clicks, but it is based on a single day of data and no ground truth. Beyond significance tests on cluster isolation and related

metrics, these results only flag suspicious activity rather than definitively identify bots. We therefore recommend the validation steps outlined in Section **??**.

- The isolated spike in TTC that guided much of our analysis could also stem from non-fraud phenomena—e.g. interface glitches or infrastructure fallback mechanisms, it should be validated further.
- Our duplication p-value signal, derived from a historical query distribution, may not capture current trends; a more robust likelihood model for query duplication by word count is needed.

7 Future work

If more time was available, under the same constraints (same dataset, no ground truth, no additional data sources) we can do the following :

- **Deeper domain knowledge:** public papers, internal user studies, etc. would be more helpful for feature engineering—to know the best signals to focus on.
- **Autoencoders:** we tried a vanilla autoencoder on the features and haven't obtained interpretable results. We believe that with careful representation learning, neural networks can be helpful here and might provide deeper insights.
- **Siamese networks:** fitting a siamese network looks promising, given that they train on pairs and output whether they are similar or dissimilar. They would not suffer from class imbalance like other models. While labels are required, we can still leverage the IForest+calibrated probabilities to extract a small subset with high confidence and explore the output of the siamese model.

References

[1] Barracuda Networks, *Bot Attacks Report, Vol. 1*, 2021.
Online

[2] Imperva, *2025 Bad Bot Report*, Imperva, 2025. https://www.imperva.com/resources/wp-content/uploads/sites/6/reports/2025-Bad-Bot-Report.pdf

[3] Muhammad Saiful Iqbal, Fatma Zulkernine, Fadi Jaafar, and Yu Gu, "FCFraud: Fighting Click-Fraud from the User Side," in *Proc. IEEE 17th Int. Symp. on High Assurance Systems Engineering (HASE)*, 2016.

[4] Kenneth C. Wilbur and Yi Zhu, "Search Advertising: Disentangling the Real Payoff," *Marketing Science*, vol. 27, no. 3, pp. 407–423, 2008. https://archive.nyu.edu/jspui/bitstream/2451/27630/2/search_advertising2.pdf

[5] Riwa Mouawi, Imad H. Elhajj, Ali Chehab, and Ayman Kayssi, "Crowdsourcing for Click Fraud Detection," *EURASIP Journal on Information Security*, vol. 2019, article 11, 2019. https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-019-0095-1

[6] Apoorva Srivastava, *Real-Time Ad Click Fraud Detection*, Master's thesis, San Jose State University, 2020. https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1916&context=etd_projects

[7] Matthieu Faou, Amandine Lemay, David Décary-Hétu, Jérôme Calvet, Frédéric Labrèche, Mathieu Jean, Benoît Dupont, and Joël Fernandez, "Follow the Traffic: Stopping Click Fraud by Disrupting the Value Chain," in *Proc. 14th Annual Conference on Privacy, Security and Trust (PST)*, Auckland, NZ, 2016, pp. 464–476. https://www.benoitdupont.net/en/publications/follow-the-traffic-stopping-click-fraud-by-disrupting-the-value-chain/

[8] Sandip Sinharay and Peter W. van Rijn, "Assessing Fit of the Lognormal Model for Response Times," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 5, pp. 534–568, 2020. https://files.eric.ed.gov/fulltext/EJ1266408.pdf

[9] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, *Isolation Forest*, in *Proc. IEEE ICDM*, 2008. PDF

[10] Wasia Ahmad, "AOL Query Log Analysis" (GitHub repository), 2017. github.com/wasiahmad/aol_query_log_analysis

[11] Scikit-learn Developers, "sklearn.ensemble.IsolationForest — Scikit-learn 1.5.2 documentation," *Scikit-learn Project*, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

[12] Stuart Coles, *An Introduction to Statistical Modeling of Extreme Values*, Springer, 2001. https://doi.org/10.1007/978-1-4471-3675-0

[13] Microsoft Advertising, *Traffic Quality Center*, Microsoft Advertising Help, 2025. https://help.ads.microsoft.com/#apex/ads/en/60223/-1

[14] Paul Embrechts, Claudia Klüppelberg, and Thomas Mikosch, *Modelling Extremal Events for Insurance and Finance*, Springer, 1997. https://doi.org/10.1007/978-3-642-33483-2

[15] b7amine, *ad-click-bot-detection*, GitHub repository, 2025. [Online]. Available: https://github.com/b7amine/ad-click-bot-detection [Accessed: Apr. 24, 2025].

"Generalized Pareto distribution," *Wikipedia*, en.wikipedia.org/wiki/Generalized_Pareto_distribution

Wikipedia contributors, "Silhouette (clustering)," *Wikipedia*, 2021. https://fr.wikipedia.org/wiki/Silhouette_(clustering)

Leland McInnes, John Healy, and James Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018. https://umap-learn.readthedocs.io/en/latest/

scholarworks.sjsu.edu/etd_projects/916

Appendix

A Feature engineering

A.1 Features descriptions

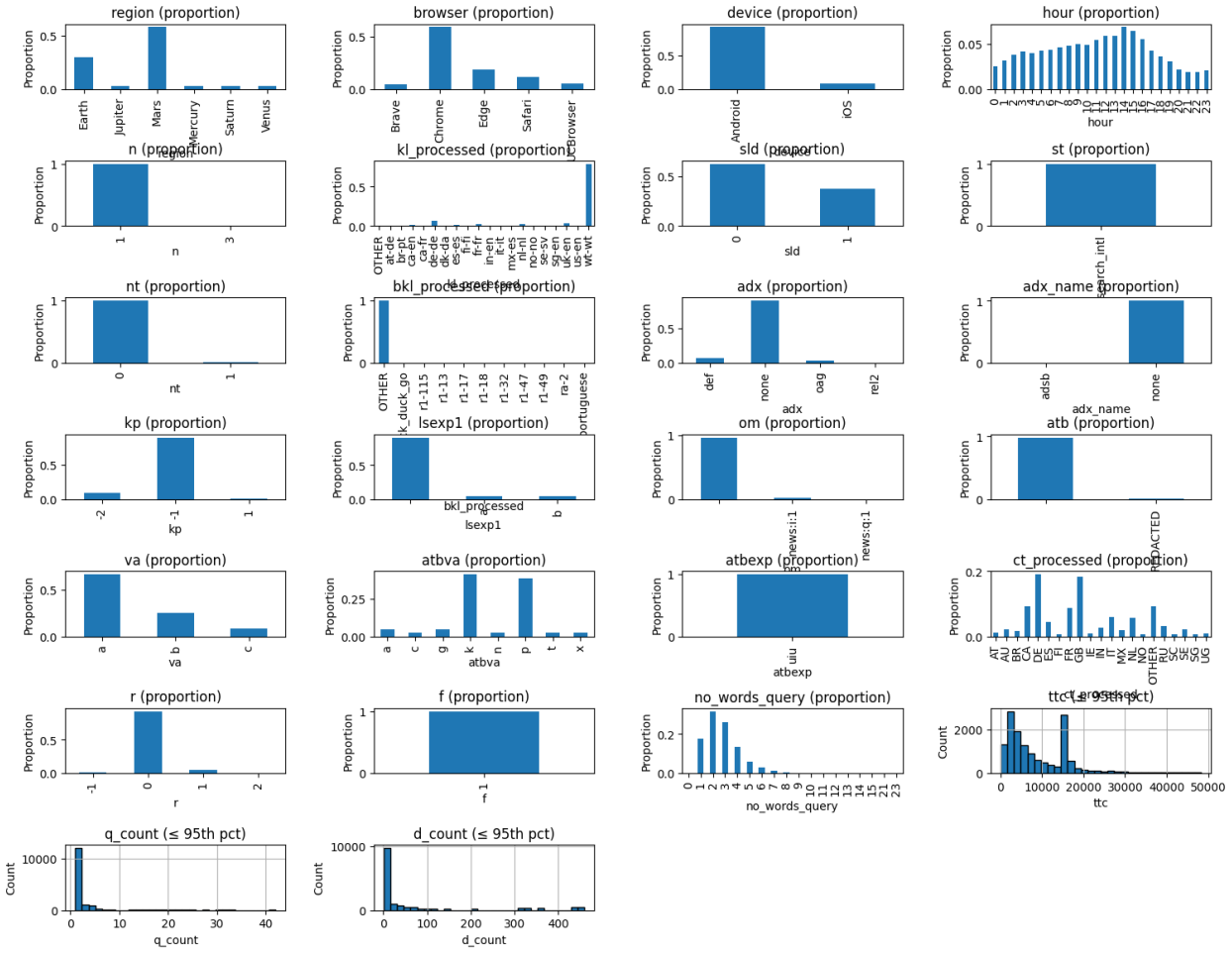


Figure 12: Distribution of Time-to-Click in Appendix.

A.2 Distributions of extracted fields

We implement the TTC–anomaly signal in four steps:

1. **Trim tails and fit.** Drop the top 1% of TTC values to avoid extreme outliers, then fit a log-normal density
$$f_{\mathrm{LN}}(t\,;s,\mathrm{loc},\mathrm{scale}) \;=\; \frac{1}{t\,s\sqrt{2\pi}}\exp\!\Big(-\tfrac{(\ln t-\ln \mathrm{scale})^2}{2s^2}\Big)$$
to the remaining data via `stats.lognorm.fit`.

2. **Bin the support.** Create 299 equal-width bins from min(TTC) to the 99th percentile, plus a final bin capturing the maximum. Denote the bin boundaries by $\{b_i\}_{i=0}^{300}$, centers $m_i = (b_i + b_{i+1})/2$, and widths $\Delta_i = b_{i+1} - b_i$.

3. **Compute densities.** For each bin i:

$$\hat{p}_i = \frac{\#\{j : b_i \le \mathrm{TTC}_j < b_{i+1}\}}{N}, \qquad p_i = f_{\mathrm{LN}}(m_i)\,\Delta_i.$$

   Then form the residual

$$\delta_i \;=\; \hat{p}_i \;-\; p_i.$$

4. **Attach to clicks.** Build a `IntervalIndex` from the $\{b_i\}$, map each click's TTC to its bin index i via `get_indexer`, and merge the corresponding δ_i back into the click-level table. The resulting Δ_{TTC} is a continuous anomaly score: large positive values mark clicks tightly clustered at delays that deviate sharply from the human log-normal pattern.

A.3 Computing the Query-Duplication p-Value

We derive a length-conditioned duplication prior from the AOL query logs and use it to assign each click a p-value:

1. **Parse and align.** Read every log file, extract (q, date) pairs, convert date to timestamps, sort globally, and keep only the last N consecutive entries to match the click dataset size.

2. **Count duplicates.** Compute

$$\mathrm{count}(q) \;=\; \#\{\text{occurrences of } q\}$$

   via `value_counts()` on the query column.

3. **Bucket query lengths.** For each query q, set

$$L(q) \;=\; \min\big(6,\ \text{number of words in } q\big),$$

   and apply the same clipping to the clicks dataset.

4. **Build quantile dictionaries.** For each $k = 1, \ldots, 6$, let $\{c_i\}$ be the duplication counts from AOL with $L(q) = k$. Compute

$$Q_k = \big\{\mathrm{quantile}(\{c_i\}; 0, 0.01, \ldots, 1)\big\},$$

   a 101-entry vector of quantiles.

5. **Compute p-values.** For a click of length k with observed count c, estimate its empirical CDF

$$\hat{F}_k(c) = \frac{1}{101} \sum_{v \in Q_k} \mathbf{1}[v \le c],$$

   and set

$$p \;=\; 1 - \hat{F}_k(c).$$

A low p-value thus signals that duplicating a given query string as often as observed is very unlikely under the AOL baseline.

B Baseline results

B.1 Grid search results

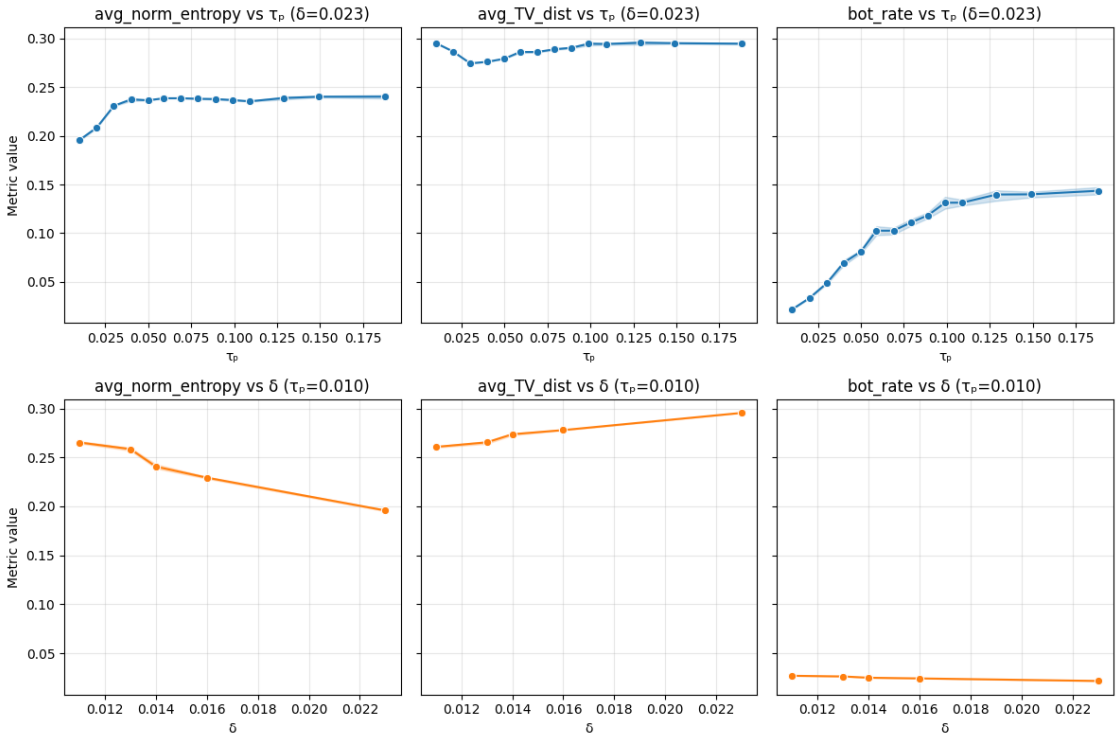


Figure 13: Grid search results on baseline signals , Individual TVs and Entropies can be explored for further insights

C ML

C.1 Grid search results : Iforest

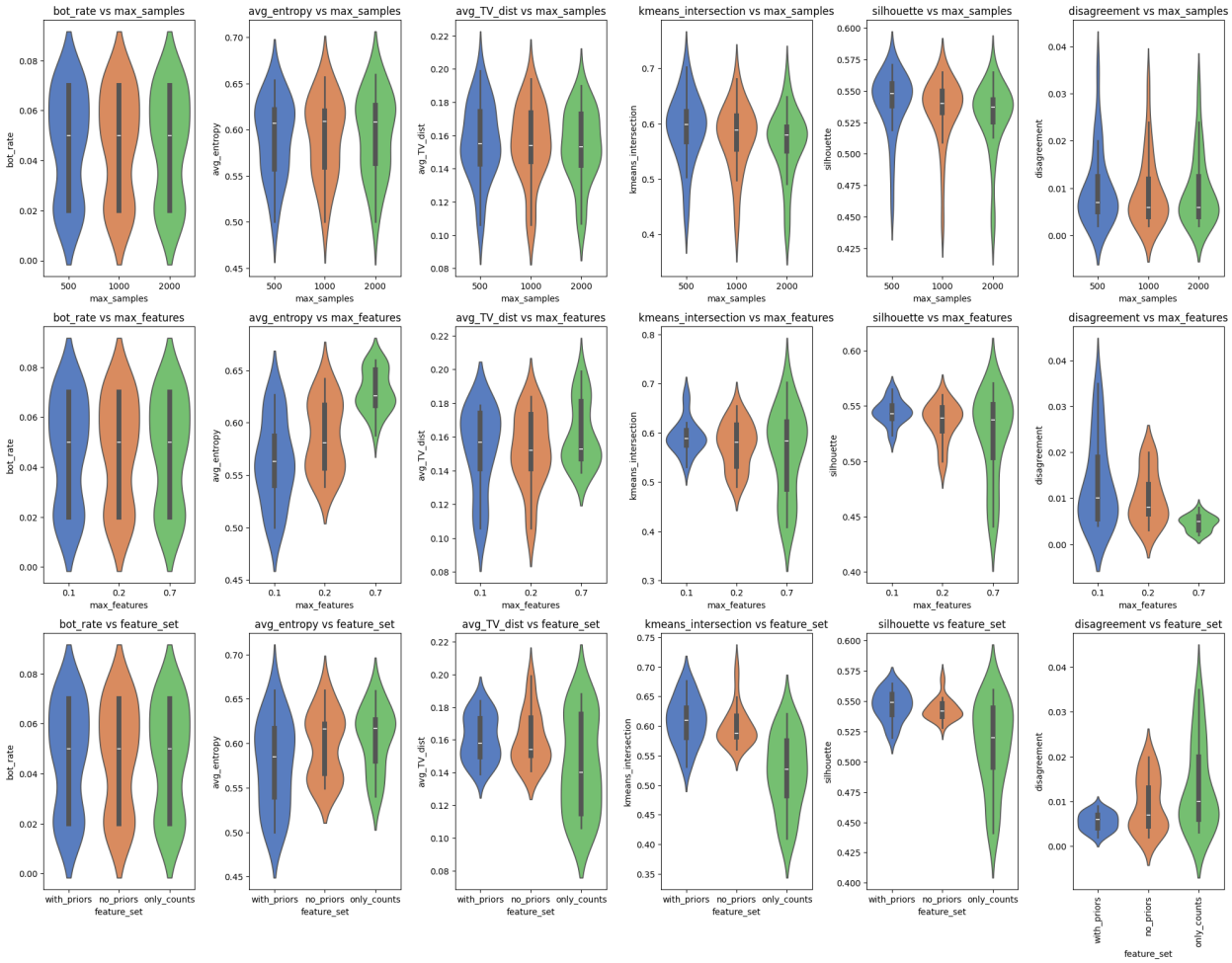


Figure 14: Grid search results on baseline signals