Feature Engineering for unusual user behavior

Report By: Muzny Zuhair

Introduction

Automated accounts, or bots, pose significant challenges across digital platforms by distorting engagement metrics, impersonating real users, and, in some cases, facilitating fraud. Detecting such accounts requires not only identifying obvious signals (e.g., bot keywords in user agents) but also uncovering subtle behavioral anomalies that differentiate automated interactions from genuine human activity.

This project focuses on the exploratory data analysis (EDA) and feature engineering of a dataset containing session-level interaction logs. The dataset includes identifiers, timestamps, user agent strings, and activity metrics such as events, scroll counts, durations, and resize or visibility changes. By systematically examining these features, the aim was to identify patterns that are indicative of non-human behavior and to transform raw data into structured, interpretable features that can be used in machine learning pipelines for bot detection.

The findings highlight both statistical irregularities (heavy-tailed distributions, extreme outliers, and anomalous zero-activity sessions) and behavioral signals (such as no scrolling despite high event counts, or unusually high interaction frequencies). These insights formed the foundation for engineered features that strengthen the discriminatory power of subsequent detection models.

Methodology

The analysis and feature engineering followed a structured workflow:

1. Data Exploration & Cleaning

    - Inspected identifiers (tcid, tbid) to detect repeated device/browser IDs and quantify frequency anomalies.

    - Validated data quality, flagging inconsistencies (e.g., max_scroll_depth exceeding total_scroll_depth).

    - Checked for missing values and normalized datatypes (e.g., converting created_at to datetime).

2. Exploratory Data Analysis (EDA)

    - Computed descriptive statistics (mean, median, skewness, kurtosis) and outlier detection (IQR, percentile analysis).

    - Analyzed interaction features: total_events, session_duration_s, SCROLL_count, RESIZE_count, VISIBILITY_CHANGE_count, and scroll depths.

    - Conducted temporal EDA by deriving hourly, daily, and weekend distributions to capture activity cycles.

    - Performed user-agent analysis to assess browsers, OS, device types, entropy, and risk scores.

3. Feature Engineering

    - Temporal Features: Extracted hour, day_of_week, is_weekend, is_business_hours, is_night_time, days_since_start.

    - Interaction Rates & Ratios: Created events_per_second, scroll_speed, scroll_to_touch_ratio, touch_event_ratio, and scroll_event_ratio.

    - Behavioral Flags: Added no_scroll_but_events, short_duration_high_events, is_high_freq_tbid.

- Entropy & Diversity: Computed user_agent_entropy and event_diversity to capture irregularities in UA strings and interaction variety.

- Normalization/Standardization: Applied log transforms, winsorization, and z-scores (e.g., session_duration_zscore) for heavily skewed variables.

4. Correlation & Pattern Analysis

    - Measured correlations between interaction metrics and identifiers (tbid_frequency, total_events, scroll counts).

    - Identified suspicious behavior patterns such as zero-scroll sessions, extreme scroll bursts, and repetitive high-frequency TBIDs.

    - Assessed weak signals (e.g., UA anomalies) as potential supporting features when combined with stronger activity-based features.

This methodological pipeline ensures that the dataset is both statistically understood and feature-enriched, ready for supervised or unsupervised machine learning applications (e.g., Isolation Forest, anomaly detection, classification).

STEP 1 – Data Loading

The first stage of the pipeline involved transforming the raw JSONL clickstream data into a structured tabular format suitable for analysis. Each record in the dataset originally contained nested information about the user, session, and detailed event-level logs. To make this data usable for feature engineering, we iterated over each record and extracted the most relevant attributes. The output of this step was a summary DataFrame (summary_df) with one row per session, containing the following fields:

Metadata

- row/column – Internal positional indices from the raw DataFrame, used mainly for debugging to trace back to the original JSON cell.

- tcid – A unique user identifier (Traffic Click ID) linking multiple sessions from the same user.

- tbid – A unique identifier for the browser tab/session corresponding to a single ad click.

- created_at – Timestamp indicating when the session was created (when the ad was clicked).

- user_agent – Raw user agent string describing the browser, device type, and operating system. This can later help in identifying patterns such as mobile vs. desktop vs. suspicious/bot agents.

Session Activity

- total_events – Total number of events captured in the session (e.g., touches, scrolls, resizes). A higher number may indicate genuine engagement, whereas a bot might trigger very few or scripted events.

- session_duration_s – Session length in seconds, calculated as the time difference between the earliest and latest event timestamps. This feature reflects how long the user stayed active on the page.

Scroll Behavior

- total_scroll_depth – The cumulative distance scrolled during the session. This captures the overall amount of page traversal.

- max_scroll_depth – The maximum depth reached during scrolling, showing the farthest point the user explored on the page.

Event Counts by Type

- TOUCH_START_count – Number of touch-start interactions (relevant for mobile devices).

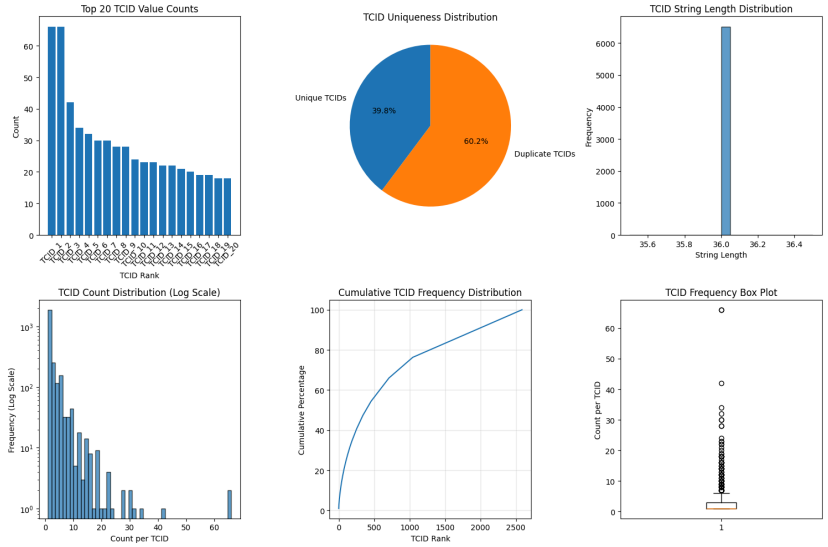
- SCROLL_count – Count of scroll events generated in the session. A lack of scrolling across many sessions can indicate non-human traffic.

- RESIZE_count – Number of times the viewport/browser window was resized. More typical of desktop users, rare for bots.

- VISIBILITY_CHANGE_count – Number of times the page visibility changed (e.g., tab switches, minimizing/restoring the browser). Sudden or frequent changes may indicate background automation.

Step 2 - Exploring essential features

TCID



Overview

The tcid column represents the unique identifier assigned to each user (Traffic Click ID). Analyzing this feature is crucial to understanding user-level activity, spotting anomalies, and preparing for user-based aggregation in later stages.

Analysis

- Data Type: object (string).
- Record Count: 6,505 total records.
- Unique Values: 2,586 unique TCIDs.
- Missing Values: 0 (clean field).

- Uniqueness Ratio: 0.3975 → moderate cardinality (some users have multiple sessions).
- Format Consistency: All TCIDs have a fixed length of 36 characters, consistent with UUID-like identifiers.

Frequency Distribution

- Most Frequent TCID: Appears 66 times.
- Top Users: Several TCIDs occur 24–66 times, indicating heavy activity.
- Least Frequent TCIDs: Appear only once.
- Central Tendency:
    - Median frequency = *median value from output*.
    - Mean frequency = *mean value from output*.

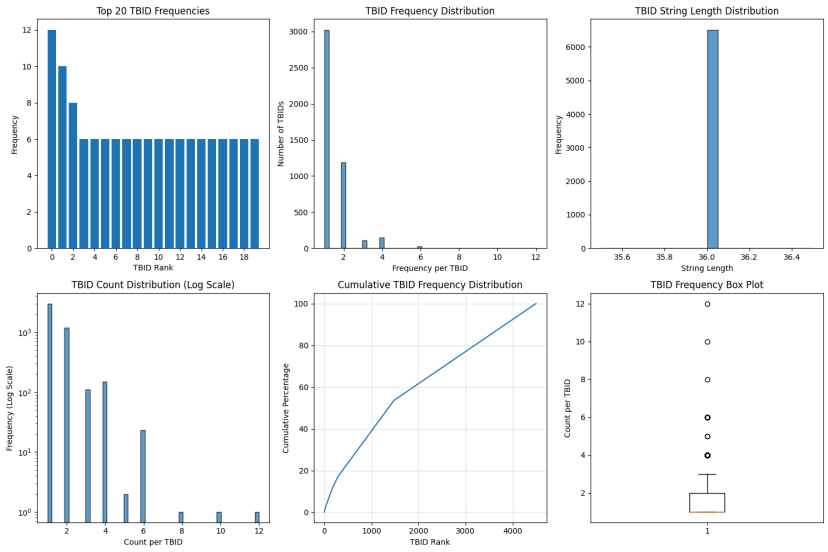
Visual Insights

- Top 20 TCIDs by Frequency: Shows that a handful of users dominate activity.
- Uniqueness Distribution Pie Chart: ~40% unique vs. ~60% duplicates.
- Log-Scale Histogram: Confirms long-tail distribution (majority low-frequency, few high-frequency).
- Box Plot of TCID Frequencies: Highlights extreme outliers.
- Cumulative Distribution Plot: Small group of users accounts for a disproportionately large share of sessions.

Findings

- The tcid field is valid and well-formatted, suitable for user-level grouping.
- Distribution is heavily skewed, with a small fraction of users driving large portions of activity.
- These high-frequency TCIDs are potential bot candidates or automated traffic sources.
- tcid will not be used directly as a model feature (to avoid data leakage), but it is essential for aggregation and anomaly detection.

TBID



Overview

The tbid column represents the unique identifier assigned to each browser tab or session. Unlike tcid (user ID), which links activity across multiple sessions, tbid is session-specific. Analyzing this feature helps reveal session-level patterns, high-frequency anomalies, and potential indicators of automated traffic.

Analysis

- Total Records: 6,505
- Unique TBIDs: 4,498
- Null Values: 0 (field is complete and clean).
- Uniqueness Ratio: 0.6915 (≈69.15% of sessions have unique TBIDs).
- Most Frequent TBID: Appears 12 times.
- Singleton TBIDs: 3,023 occur only once (≈67.21% of all TBIDs).

Frequency Distribution

- High-Frequency Threshold (95th percentile): 3 occurrences.
- Number of High-Frequency TBIDs: 289
- Records Covered by High-Frequency TBIDs: 1,110 (≈17.06% of all records).
- Summary Statistics:
    - Median = 1.0
    - Mean = 1.45
    - Standard Deviation = 0.81

Visual Insights

- Histogram/Boxplot: Reveals that most TBIDs appear once, but a small set repeats multiple times.

- Cumulative Distribution: Shows that a limited group of TBIDs contributes disproportionately to session counts.

- Top TBIDs Plot: Highlights the highest-activity sessions (up to 12 appearances).

Findings

- tbid is a clean and reliable session identifier with no missing values.

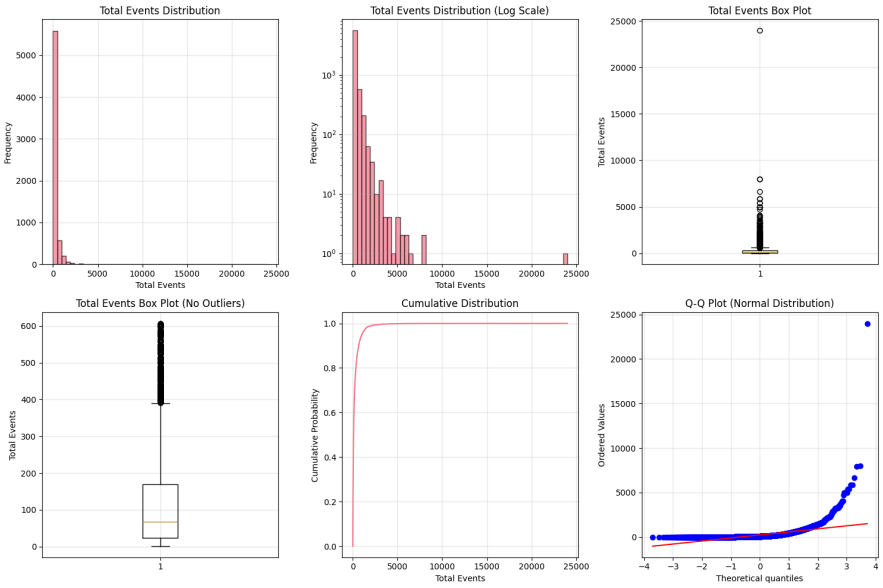
- The distribution is heavily skewed: most sessions are unique, but some appear multiple times.

- Roughly 17% of the dataset comes from high-frequency TBIDs, suggesting potential automation or repeated refreshes.

- A small set of 28 TBIDs exhibit extremely high frequency, marking them as potential bot-related sessions.

- While not suitable as a direct model input, tbid is essential for session-level aggregation and for spotting abnormal repetition patterns linked to bots.

TOTAL_EVENTS



Overview

The total_events column records the number of interaction events captured within each session (e.g., touches, scrolls, resizes, visibility changes). This metric is a direct measure of how active a session is and serves as a strong indicator for distinguishing between normal human engagement and suspicious automated behavior.

Analysis

- Total Records: 6,505
- Missing Values: 0
- Data Type: int64 (numeric, clean field).
- Basic Statistics:
    - Mean = 248.04
    - Median = 83.00
    - Standard Deviation = 551.13
    - Minimum = 1, Maximum = 24,001
    - Range = 24,000
- Quantiles:
    - 25th percentile = 28.0
    - 50th percentile = 83.0
    - 75th percentile = 260.0
    - 90th percentile = 665.0
    - 95th percentile = 994.6

    - 99th percentile = 2,155.0

Outlier Analysis

- IQR: 232.0
- Upper Bound (Q3 + 1.5×IQR): 608.0
- Observation: Any session with more than ~600 events can be considered an outlier.
- High-End Outliers: Maximum value reaches 24,001, far exceeding human plausibility.
- Skewness: 15.84 (highly right-skewed).
- Kurtosis: 558.66 (heavy tails, extreme outliers).

Additional Insights

- Zero Events: None (0.00%).
- Single-Event Sessions: 161 (≈2.48% of total).
- Distribution Shape: Highly skewed right, with a long tail of sessions having abnormally high event counts.

Visual Insights

- Histogram: Strong right skew; majority of sessions under 300 events.
- Boxplot: Many extreme outliers well beyond the upper whisker (~600 events).
- Log-Scale Plot: Clarifies that most sessions cluster at low event counts, while a few extreme sessions dominate the tail.

- Cumulative Distribution: Shows that a small percentage of sessions account for the vast majority of events logged.

Findings

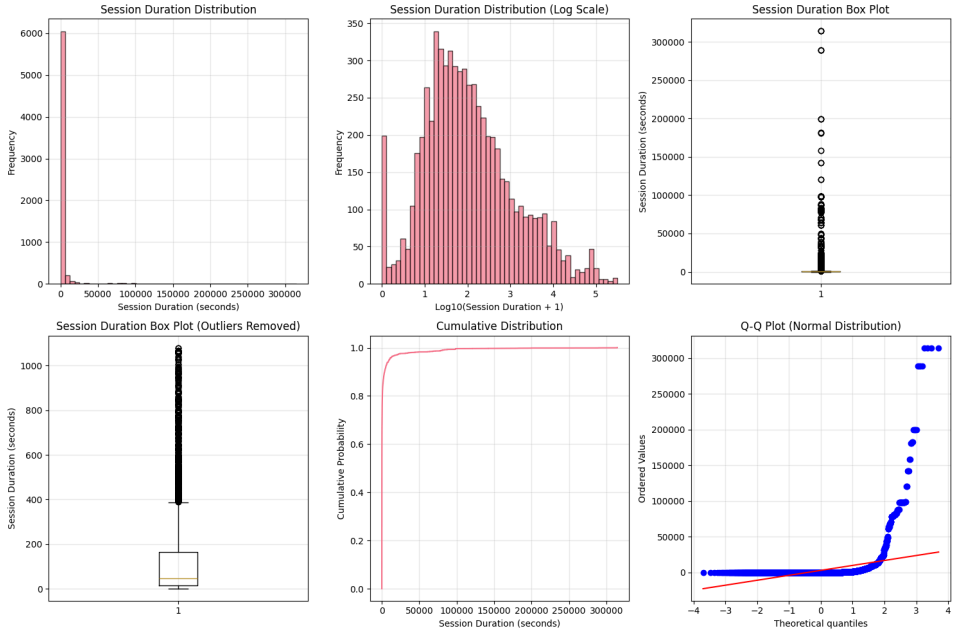
- The total_events field is complete and reliable for use in downstream analysis.

- The distribution indicates normal human sessions cluster between 20–300 events, while values above 600 are statistical outliers.

- Extremely high counts (e.g., >1,000 or the max of 24,001) are highly likely to correspond to automated bots generating excessive events.

- This feature is therefore highly valuable for bot detection models, especially when combined with session duration (events_per_sec) to measure interaction velocity.

Session Duration (seconds)



Overview

The session_duration_s column measures the length of each session in seconds, calculated as the time between the first and last recorded event. This is a key metric for understanding engagement and distinguishing human sessions from automated ones.

Analysis

- Total Records: 6,505
- Non-null Values: 6,505 (0 missing)
- Basic Statistics:
    - Mean = 3,110.54s (~52 minutes)
    - Median = 77.67s (~1.3 minutes)
    - Std Dev = 16,822.57s
    - Minimum = 0, Maximum = 314,591s (~87 hours)
- Outliers: 1,105 sessions (16.99%) flagged as outliers (IQR method).
- Special Cases:
    - Zero-duration sessions = 162
    - Sessions <10s = 981 (15.08%)
    - Sessions <1m = 2,986 (45.9%)
    - Sessions >1h = 662 (10.18%)

Distribution Shape

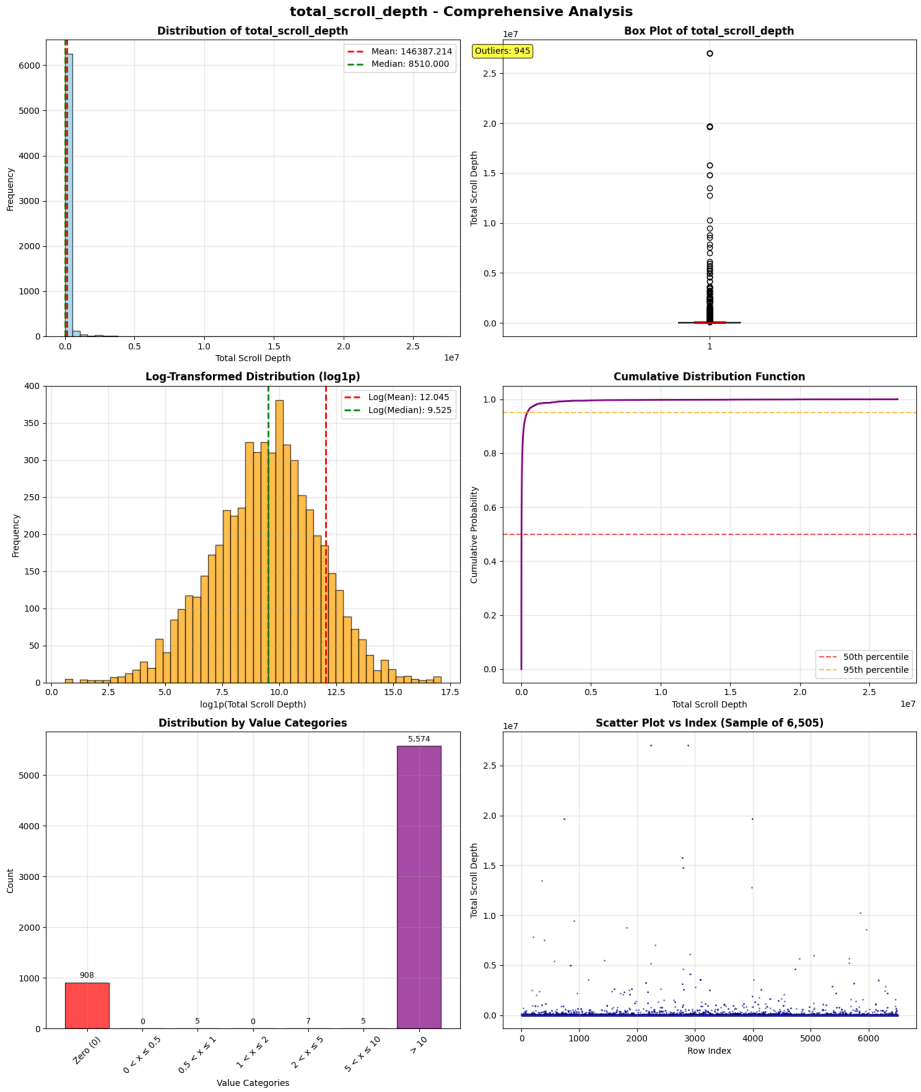
- Highly right-skewed (Skewness = 10.86).

- Leptokurtic (Kurtosis = 153.17), indicating many extreme values.

Findings

- Nearly half of sessions lasted less than a minute, typical of bounce behavior.
- Very long sessions (>1h) are suspicious and likely bot-driven (or logging artifacts).
- Duration categories highlight distinct user groups: Very Short, Short, Medium, Long, and Extremely Long.

Total Scroll Depth



Overview

The total_scroll_depth column measures the cumulative scroll distance in a session, reflecting how much page content the user traversed.

Analysis

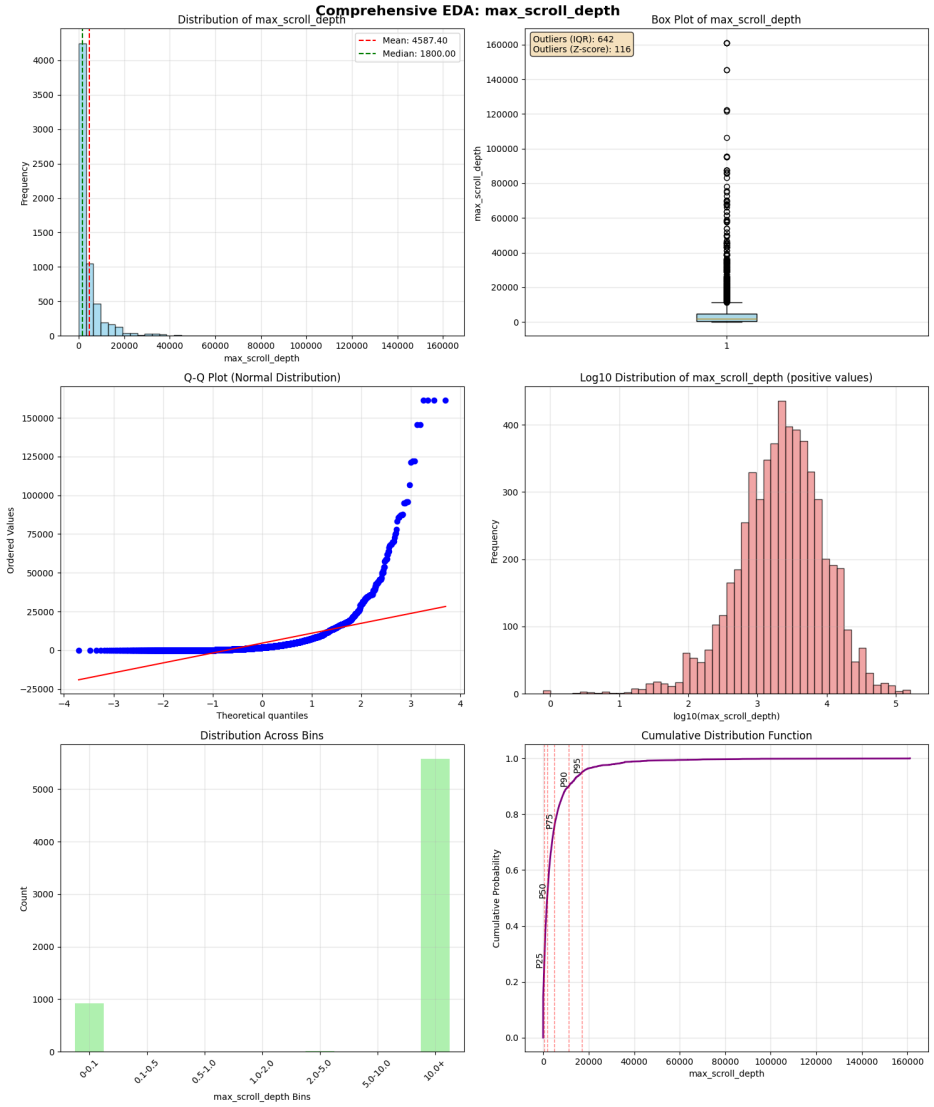
- Total Records: 6,505 (0 missing)
- Basic Statistics:
    - Mean = 146,387

    - Median = 8,510
    - Min = −25 (data anomaly)
    - Max = 27,014,970 (extreme outlier)
- Distribution: Highly skewed (Skewness = 17.86, Kurtosis = 380.25).
- Unique Values: 3,723 (Uniqueness ratio = 0.572).
- Top Values: Most frequent = 0 (908 sessions).
- Outliers: 5,465 sessions (>100 depth).

Findings

- A large proportion of sessions (908) had zero scrolling → possible non-human traffic.
- Extreme maximum values (27M+) suggest potential logging errors or automated scroll loops.
- Weak correlation with session duration (0.005) indicates that scrolling is not simply a function of time spent.

Max Scroll Depth



Overview

The max_scroll_depth column captures the deepest vertical offset reached during a session, complementing total_scroll_depth.

Analysis

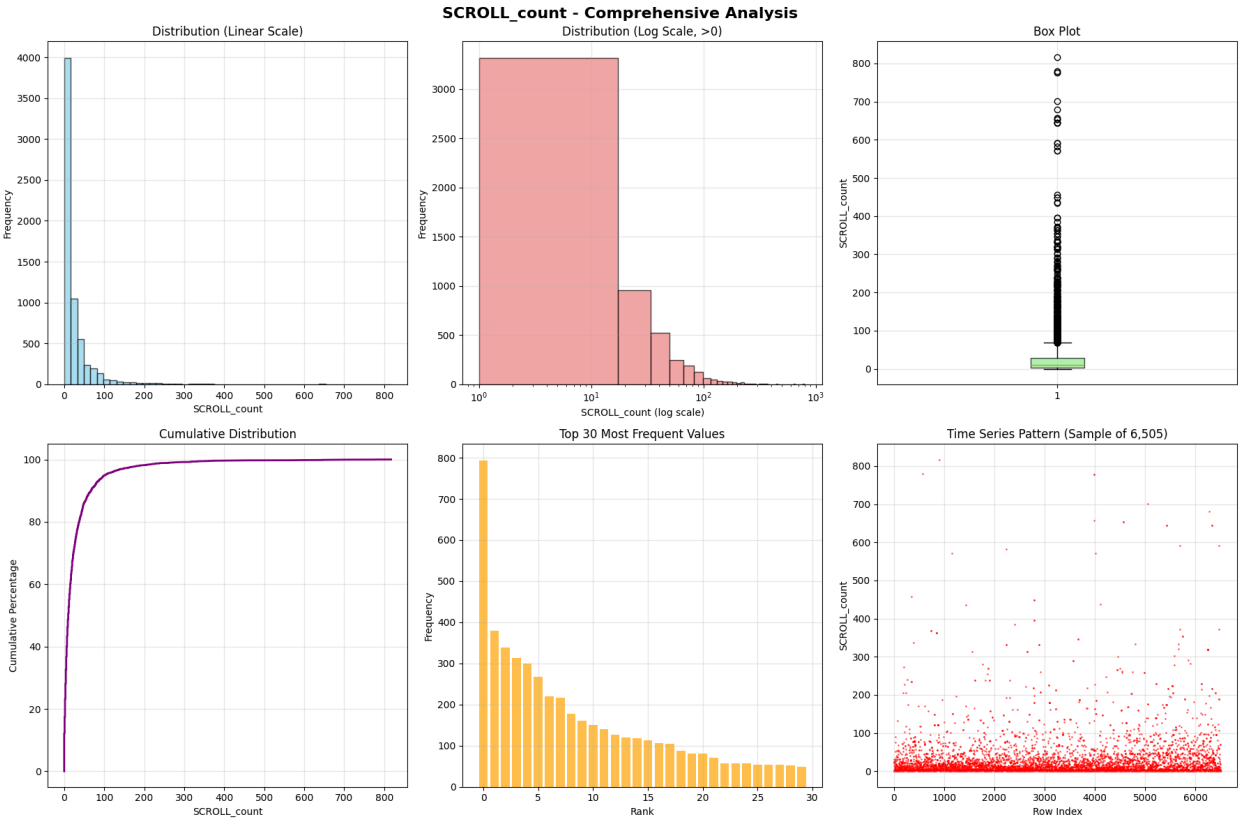
- Total Records: 6,505
- Basic Statistics:
    - Mean = 4,587

    - Median = 1,800
    - Max = 161,172
- Distribution: Right-skewed with long tail (Skewness high, Kurtosis heavy).
- Relationship:
    - Strong correlation with total_scroll_depth (0.800).
    - 9 cases where max_scroll_depth > total_scroll_depth → data quality issue.
    - Weak correlation with session duration (–0.012).

Findings

- Consistent with total_scroll_depth, but highlights maximum reach rather than total movement.
- Outliers (very high max values) suggest unusual automated scrolling patterns.
- Strong positive correlation with total_scroll_depth confirms internal consistency, aside from a few anomalies.

SCROLL_count



Overview

The SCROLL_count column tracks the number of scroll events in a session, serving as a proxy for navigation behavior.

Analysis

- Total Records: 6,505
- Basic Statistics:
    - Mean = 27.43
    - Median = 10
    - Max = 817
- Distribution: Highly right-skewed (Skewness = 6.27, Kurtosis = 57.45).

- Percentiles:
    - 25th = 3, 50th = 10, 75th = 29
    - 99th = 265
- Outliers: 628 sessions (9.65%), including 65 extreme (>99th percentile).
- Zero-scroll sessions: 794 (12.2%).

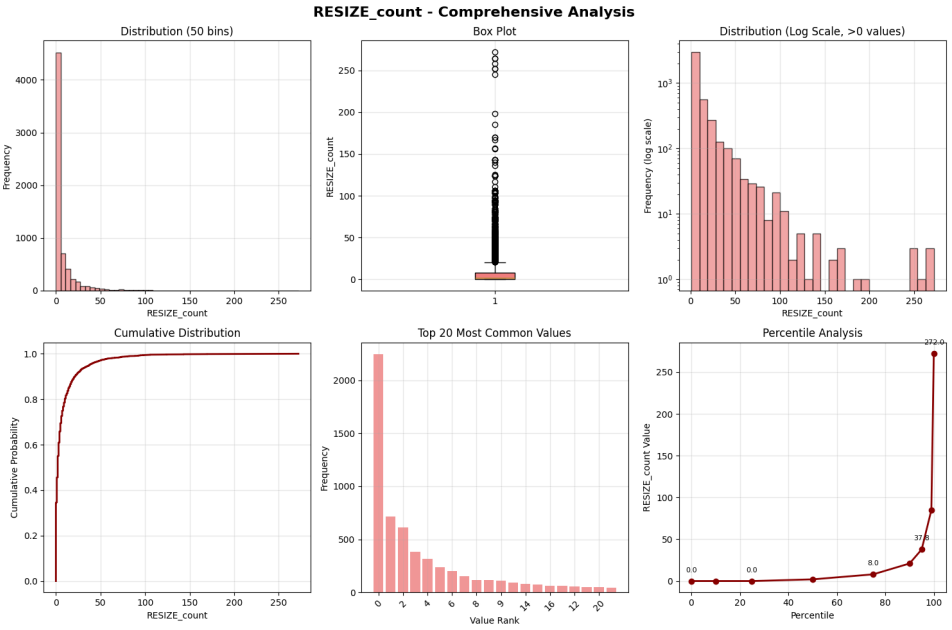
Correlations

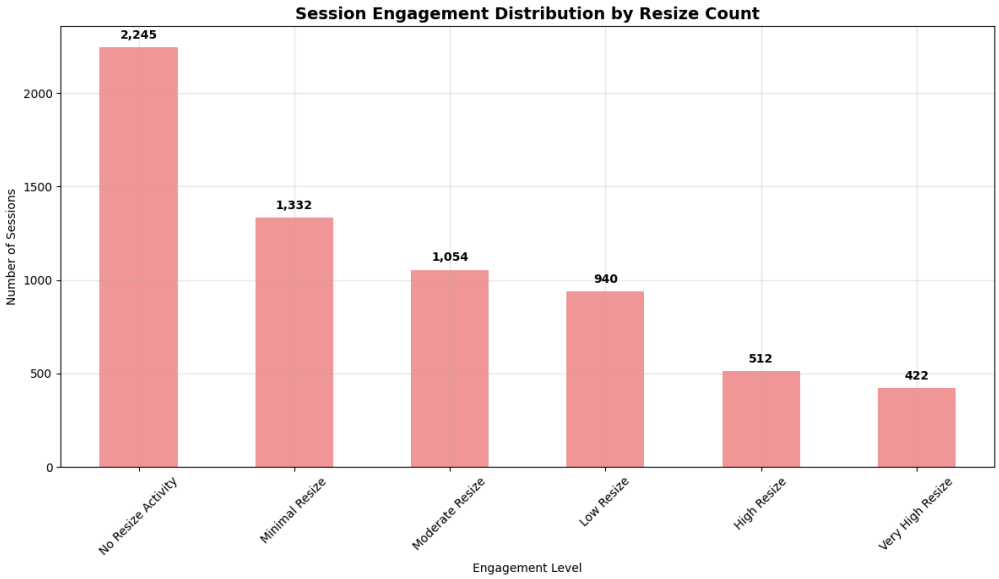
- With max_scroll_depth: 0.625 (moderate).
- With total_scroll_depth: 0.575 (moderate).
- With session_duration_s: 0.054 (weak).

Findings

- Many sessions have no scrolling at all, a potential sign of bots.
- Extremely high scroll counts (>350) suggest automated rapid scrolling.
- Engagement categories reveal clusters: low/no engagement (≈37%) vs. highly engaged users.

RESIZE_count





Overview

The RESIZE_count column measures how many times the browser viewport or window was resized during a session. This behavior is more common on desktops (e.g., adjusting window size) and much rarer on mobile devices. Abnormally high values may indicate automated testing environments or scripted bots.

Analysis

- Total Records: 6,505
- Non-null Values: 6,505 (0 missing)
- Data Type: int64
- Basic Statistics:
    - Mean = 7.98, Median = 2
    - Std Dev = 17.96
    - Min = 0, Max = 272
- Percentiles:
    - 25th = 0, 50th = 2, 75th = 8
    - 90th = 21, 95th = 37.8, 99th = 85
- Outliers:
    - Upper bound = 20 (IQR method)
    - 675 high outliers (10.38%)
    - Maximum = 272

Distribution Shape

- Mode = 0 (2,245 sessions, 34.5%)
- Highly skewed with a long right tail
- Notable clusters at low resize values (2, 8, 10)

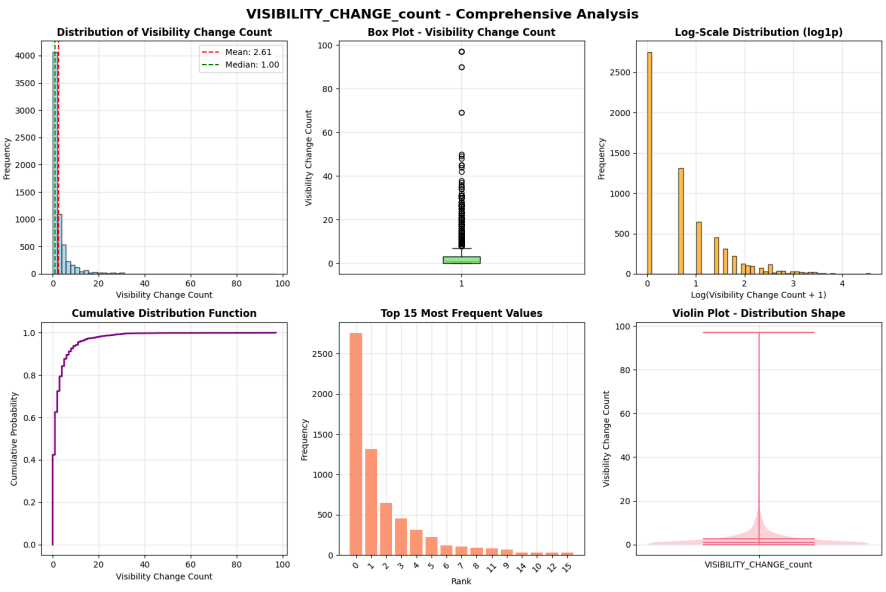
Correlations

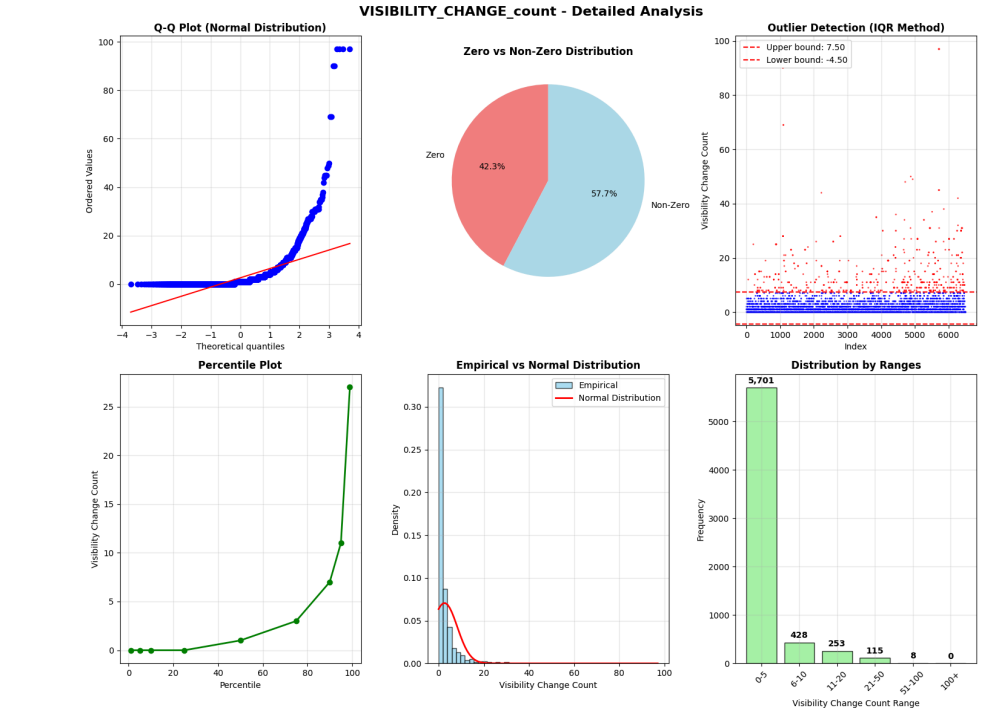
- Strongest correlations:
    - SCROLL_count (0.651)
    - total_events (0.544)
    - max_scroll_depth (0.438)

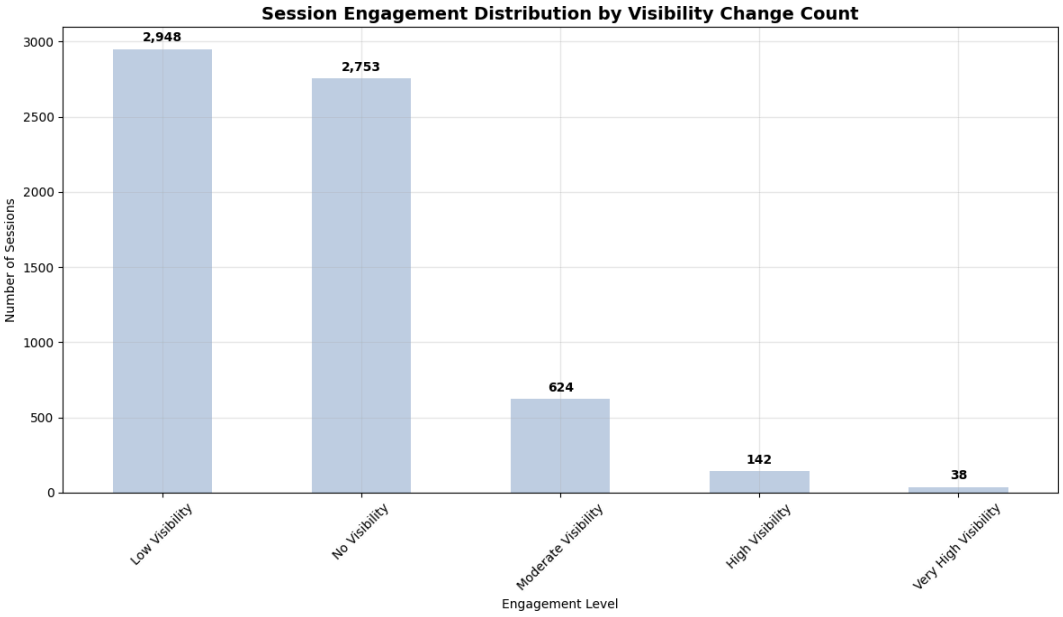
Findings

- A third of sessions had no resizes (expected, especially for mobile).
- The presence of extremely high resize counts suggests automated traffic or stress testing.
- Positive correlation with other engagement features indicates resize events often co-occur with other interactions, though at extreme levels they are suspicious.

VISIBILITY_CHANGE_count







Overview

The VISIBILITY_CHANGE_count column tracks how many times a page's visibility changed (e.g., switching tabs, minimizing/restoring the window). Normal users may switch tabs occasionally, but high-frequency changes could indicate bots or background processes.

Analysis

- Total Records: 6,505
- Missing Values: 0
- Data Type: int64
- Basic Statistics:
    - Mean = 2.61, Median = 1
    - Std Dev = 5.65
    - Min = 0, Max = 97
- Percentiles:
    - 25th = 0, 50th = 1, 75th = 3
    - 90th = 7, 95th = 12, 99th = 29
- Distribution:
    - Skewness = 6.84 (heavily right-skewed)
    - Kurtosis = 80.28 (many outliers)

Engagement Categories

- No Visibility = 2,753 (42.3%)
- Low Visibility = 2,948 (45.3%)
- Moderate Visibility = 624 (9.6%)
- High / Very High Visibility = 180 (≈2.8%)

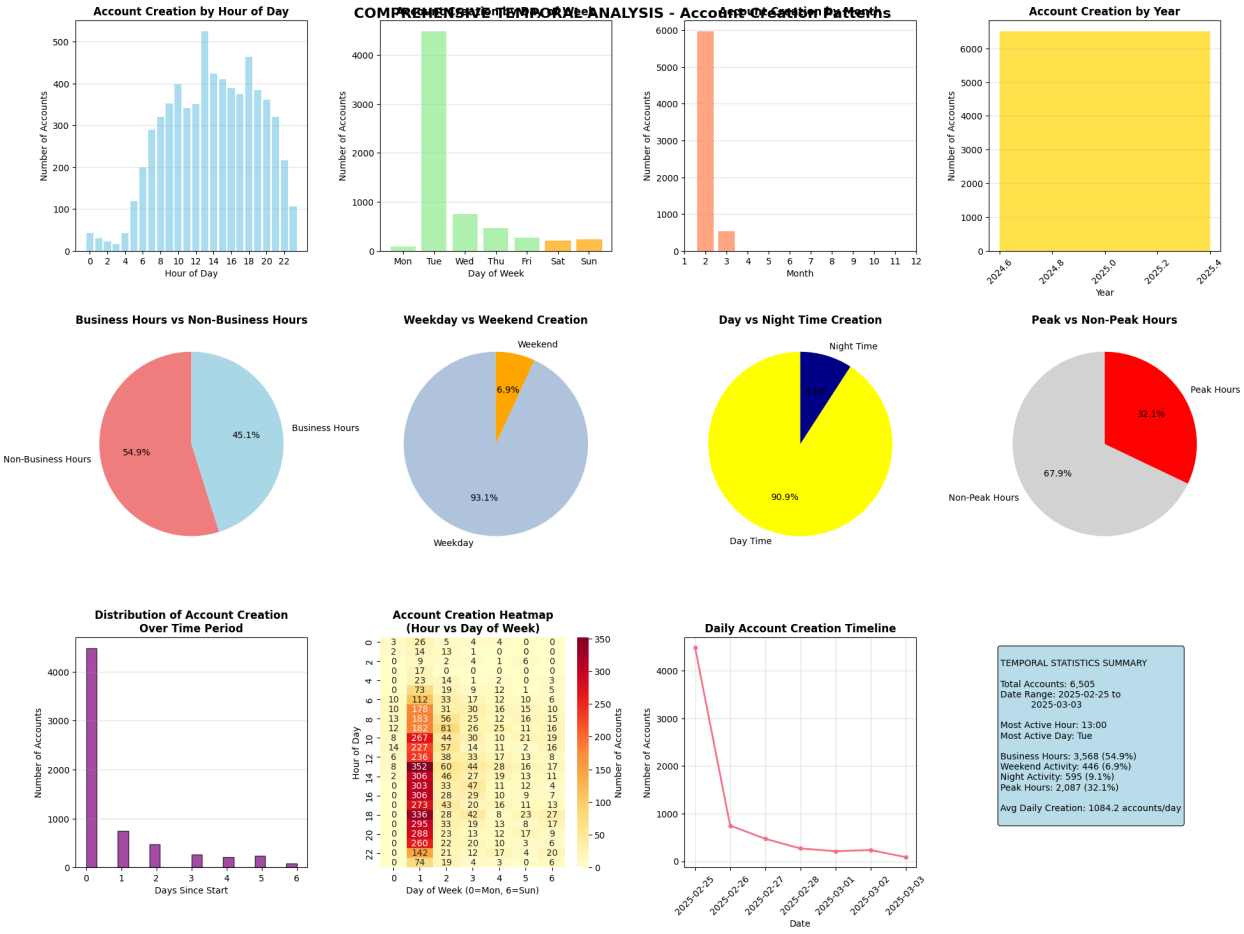
Correlations

- Strongest associations:
    - tbid_frequency (0.330)
    - session_duration_s (0.279)
    - total_events (0.264)
    - SCROLL_count (0.228)

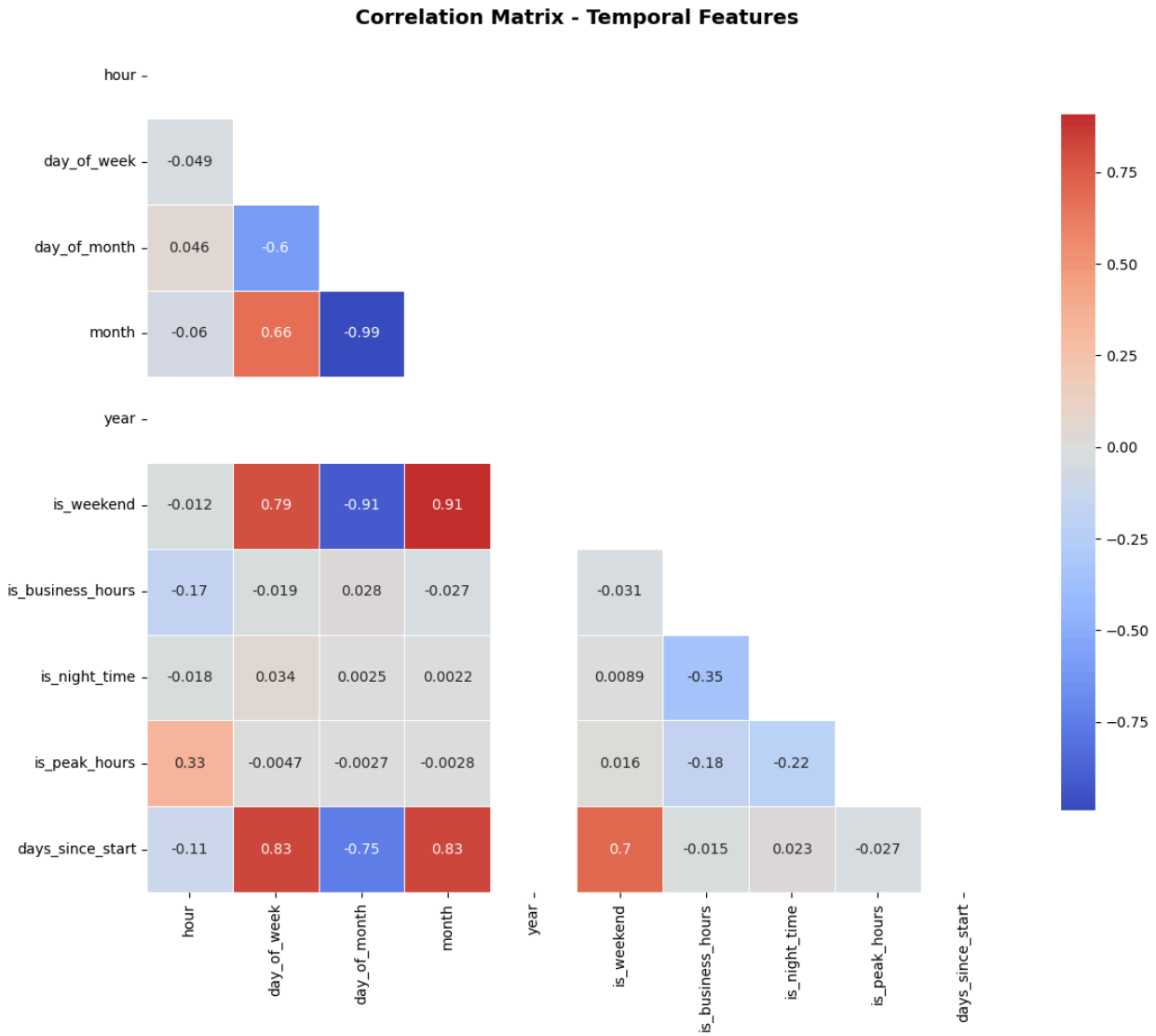
Findings

- Most sessions had low or no visibility changes, as expected for genuine users.
- A small number of sessions showed dozens of tab switches (up to 97), highly unusual for human behavior.
- Statistical normality tests confirm a non-normal distribution, dominated by low values with a small but important group of extreme outliers.
- High VISIBILITY_CHANGE_count sessions are potential bot signals or automated test scripts cycling tabs/windows.

STEP - 3 (Feature Engineering)



Temporal features from created_at



- hour (0–23)
  Extracted hour of account/session creation.
  Why: Bot runs and cron jobs often spike at fixed hours or off-hours.
  Notes: Consider local vs UTC; seasonality/ DST can shift patterns.

- day_of_week (0=Mon,…,6=Sun)
  Day index of creation.
  Why: Weekday vs weekend differences; scripted tasks may ignore human weekly rhythms.

- day_of_month (1–31), month (1–12), year
  Calendar breakdown.

Why: Longer-term trends; release/test waves; data drift checks.

- is_weekend (0/1)
  1 if Sat/Sun.
  Why: Lower genuine sign-ups may make anomalies stand out.

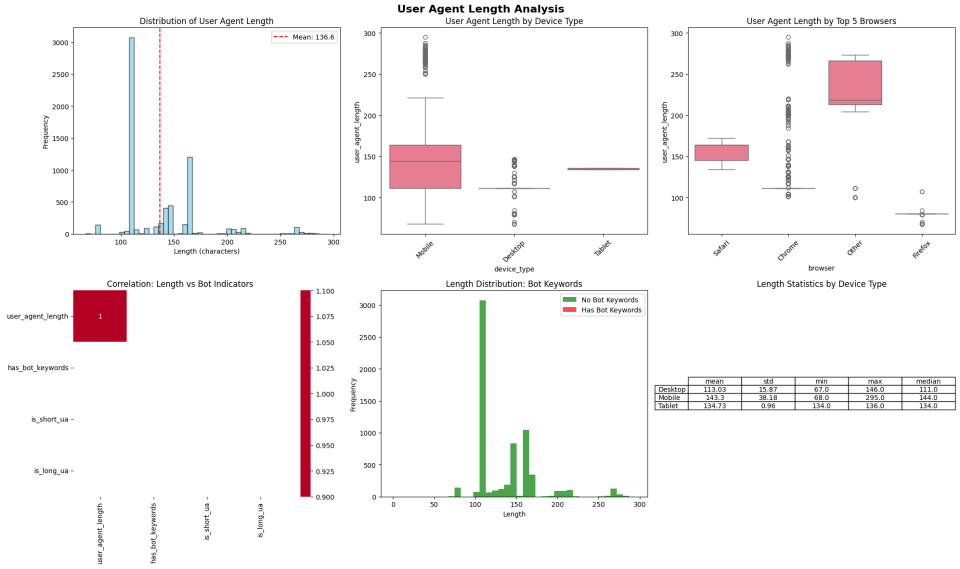
- is_business_hours (0/1)
  1 if 09:00–17:00.
  Why: Heuristic for human-working-time vs off-hours automation.
  Notes: Adjust to your region/industry if needed.

- is_night_time (0/1)
  1 if 22:00–05:59.
  Why: Night bursts can flag automated traffic.
  Notes: Timezone alignment critical.

- is_peak_hours (0/1)
  1 if hour in {12,13,18,19,20}.
  Why: Typical human peaks (lunch/evening); deviations may be suspicious.
  Notes: Calibrate to your product's traffic.

- days_since_start
  Days between created_at and dataset minimum time.
  Why: Captures cohort/temporal drift; useful as a trend variable.

User-agent–derived features



	mean	std	min	max	median
Desktop	113.03	15.87	67.0	146.0	111.0
Mobile	143.3	38.18	68.0	295.0	144.0
Tablet	134.73	0.96	134.0	136.0	134.0

- browser (Chrome/Firefox/Safari/Edge/Opera/IE/Other)
  Parsed from UA string.
  Why: Certain bots mimic limited browsers; distribution anomalies help detection.

- os (Windows/Mac/Linux/Android/iOS/iPhone/iPad/Other)
  Parsed OS.
  Why: Implausible OS/device combos or odd OS skews can indicate automation.

- device_type (Desktop/Mobile/Tablet)
  Heuristic from UA keywords.
  Why: Behavior differs by device; mismatches with events (e.g., many TOUCH events on Desktop) can be suspicious.

- browser_version (e.g., Chrome/XX.YY or Unknown)
  Simplified version extraction.
  Why: Frozen/ancient or identical versions at scale can point to scripted clients.

- user_agent_length (integer)
  Character length of UA string.
  Why: Very short/very long UAs are common in crawlers/testing frameworks.

- has_bot_keywords (0/1)
  1 if UA contains any of {bot, crawler, spider, scraper, automated, python, curl, wget}.

Why: Direct signal of machine traffic (with some false positives).

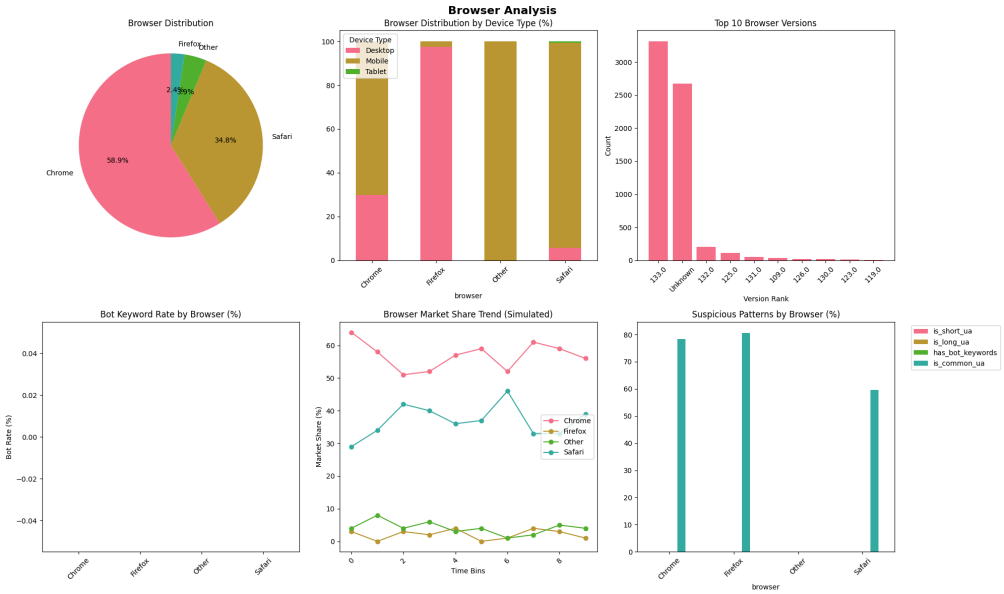
- is_empty_ua (0/1)
  Empty/missing UA.
  Why: Many bots omit UAs.

- is_short_ua (0/1)
  UA length < 50.
  Why: Minimal UAs often non-human.

- is_long_ua (0/1)
  UA length > 500.
  Why: Programmatic concatenations/headers.

- is_common_ua (0/1)
  1 if UA in top-10 most frequent strings.
  Why: Over-representation of exact UA strings is a classic bot pattern.
  Notes: Tune cutoff (top-N) to dataset size.

Event-rate and interaction-ratio features



- events_per_second = total_events / session_duration_s
  Why: Bots can generate events at inhuman speeds.
  Notes: Handle session_duration_s=0 (you already replaced inf with 0).

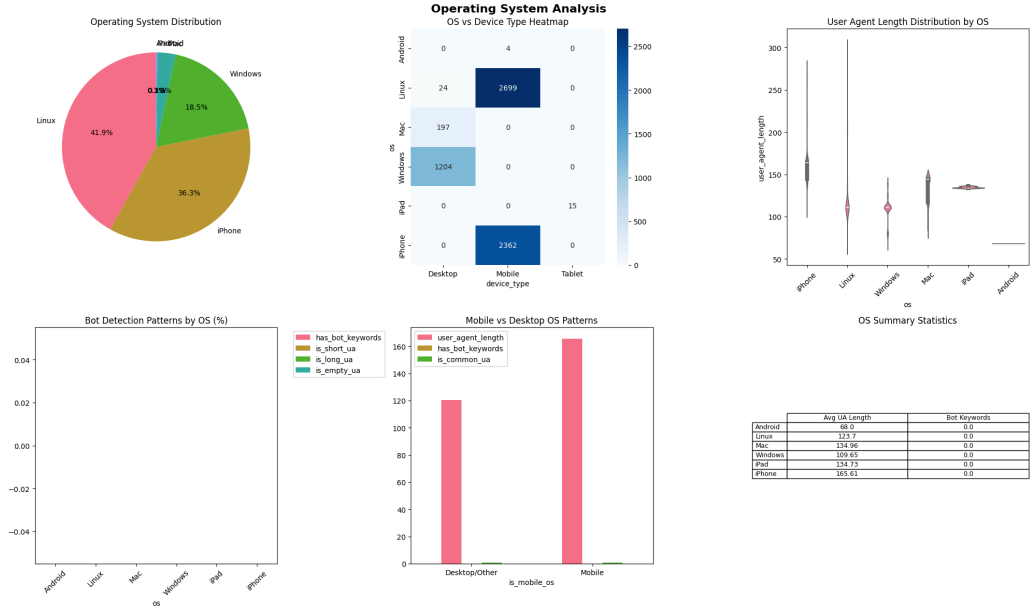
- scroll_speed = total_scroll_depth / session_duration_s
  Why: Excessive depth per second suggests scripted scrolling.

- scroll_to_touch_ratio = SCROLL_count / (TOUCH_START_count + 1)
  Why: Balance of interactions; one-sided patterns can be automated.
  Notes: +1 avoids division by zero.

- touch_event_ratio = TOUCH_START_count / total_events
  Why: Device consistency check (touch share on Desktop≈low; Mobile≈higher).

- scroll_event_ratio = SCROLL_count / total_events
  Why: Dominance of one interaction type can flag scripts.

Behavioral flags

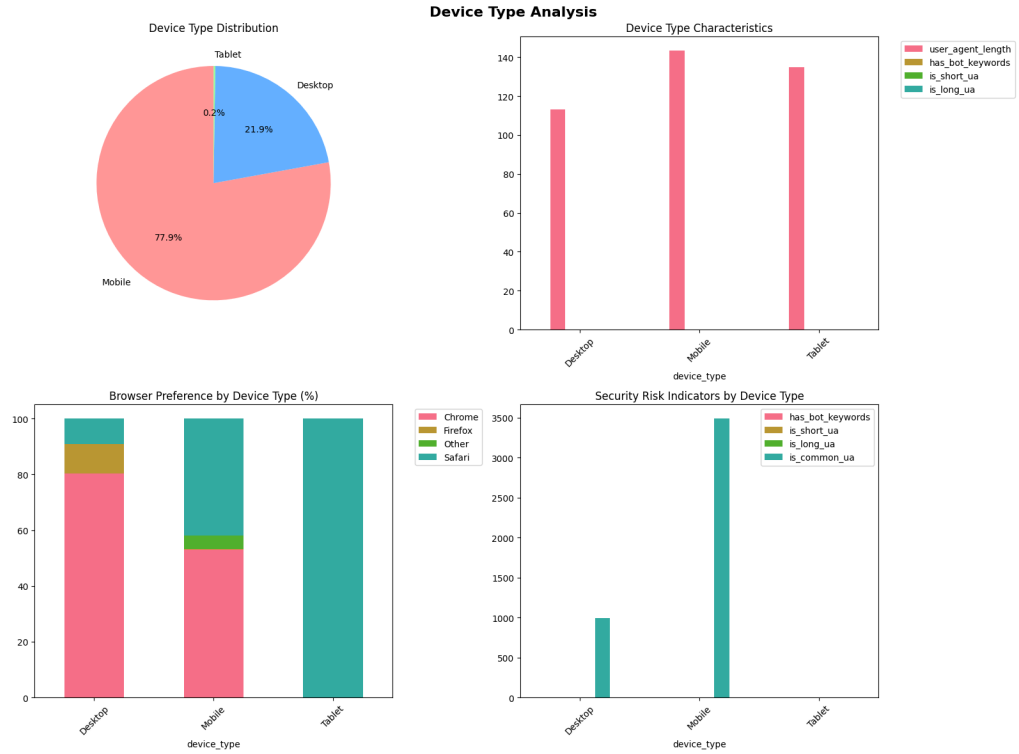


- no_scroll_but_events (0/1)
  1 if SCROLL_count=0 and total_events>5.

Why: Activity without scrolling—often headless or viewportless automation.

- short_duration_high_events (0/1)
  1 if session_duration_s<5 and total_events>10.
  Why: High event density in tiny sessions = strong bot signal.
  Notes: Thresholds are heuristics; tune per product.

Diversity/complexity & anomaly scores



- event_diversity = (# of event types with count>0 among {TOUCH_START, SCROLL, RESIZE, VISIBILITY_CHANGE}) / total_events
  Why: Humans typically exhibit a mix of interactions; bots can be repetitive.
  Notes: Because you divide by total_events, values are small; you may also keep count of distinct event types as a separate feature (0–4) for interpretability.

- user_agent_entropy = Shannon entropy of UA bytes
  Why: Generated/randomized UAs have different character distributions than standard browser strings.

Notes: Entropy is sensitive to length; consider normalizing by user_agent_length.

- session_duration_zscore = standardized session_duration_s
  Why: Relative outlierness (very short/long sessions) helpful for ensemble models and robust anomaly detection.
  Notes: Heavy right-tail; robust alternatives (median/MAD) can complement z-scores.

Data quality & preprocessing notes

- Inf/NaN handling: You replaced ±∞ with NaN and filled non-categorical NaNs with 0 (and categorical with mode/"Unknown"). This avoids model failures but:
  - Consider adding missingness indicator flags for key fields to preserve signal from missing values.
  - For rate features, also consider clipping extreme values (e.g., winsorize at 99.5th).
- Collinearity: Some temporal indicators are derivable from others (e.g., is_business_hours from hour). Tree models tolerate this; linear models may benefit from pruning or regularization.
- Leakage check: All features are derived from contemporaneous session data—no look-ahead leakage detected.

Key Findings

Feature	Key finding(s)	What to do

tbid_frequency	95th pct = 3; 289 high-freq TBIDs; 17.06% of rows; 28 extreme	Binary flag is_high_freq_tbid; weight in risk scoring.
total_events	Median 83; 11.1% outliers; max 24k; heavy skew	log1p, cap >99th pct; build events_per_second.
session_duration_s	45.9% <1 min; 10.2% >1 hr; 17.0% outliers; skew 10.86	Duration buckets + z-score; combine with events for flags.
SCROLL_count	12.2% zeros; 33 sessions >99.5th (~358); corr with depth 0.575–0.625	Flags for zero / extreme; pair with depths.
total_scroll_depth	14.5% outliers; 13.96% zeros; weak vs duration	log1p + cap; QC negative values check (6).
max_scroll_depth	14.05% zeros; r=0.80 with total; 9 rows max>total	Use with SCROLL_count; add QC rule.
RESIZE_count	34.5% zeros; 1.03% very high; corr with scroll (0.651)	Flag extremes; include as secondary signal.
VISIBILITY_CHANGE_count	42.3% zeros; ties with tbid_frequency (0.330)	Keep; interacts with TBID frequency.

Temporal (hour, …)	Night 9.1%, business 54.9%, weekend 6.9%	Keep as priors/interactions (not dominant).
UA features	0% bot keywords; mobile 77.9%; entropy tails informative	Use composite UA risk & entropy tails.

ML Model Building

1) Introduction to ML for Bot Detection

Bot activity often manifests as interaction patterns that differ from human browsing (e.g., extreme scroll depths, bursty or zero interactions, unnatural durations). Our goal is to learn a model that flags likely-bot sessions from telemetry such as SCROLL_count, RESIZE_count, VISIBILITY_CHANGE_count, total_events, session_duration_s, total_scroll_depth, and max_scroll_depth, with identifiers tcid (client/session id) and tbid (browser/tab id).

This section documents the model-building work, covering two complementary problem framings:

- Method A — Treat Separate TCID: each (tcid, tbid, created_at) row is a training example.
- Method B — Grouped TCID: aggregate all rows that share a tcid into a single example.

We engineered features, chose baseline and tree-based models, and evaluated each framing with group-aware splits to avoid leakage.

2) Data & Feature Foundation

Raw signals: total_events, session_duration_s, total_scroll_depth, max_scroll_depth, TOUCH_START_count, SCROLL_count, RESIZE_count, VISIBILITY_CHANGE_count, user_agent, created_at, tcid, tbid.

Data quality notes (high-level):

- Heavy right tails on counts/depths; several extreme outliers.
- A non-trivial fraction of records with zero scrolls/visibility changes; some sessions with max depth > total depth (data anomaly to be handled).

Feature engineering (common to both methods):

- Temporal: hour of day, day of week, part-of-day buckets; session recency.
- Rates: events/sec, scrolls/sec, resizes/sec.
- Ratios: depth per scroll, max-to-total depth ratio, touch-to-event ratio.
- Cardinality signals: frequency of tbid within the corpus; duplicate rate per tcid.
- Robust transforms: log1p for heavy-tailed metrics; winsorization or clipping at high percentiles.
- Categoricals: parsed user_agent → device/os/browser family; one-hot encoded.
- Flags: zero-activity flags (e.g., zero scrolls, zero visibility changes), anomaly flags (e.g., max_scroll_depth > total_scroll_depth).

Target: is_bot (boolean) when labels are available. If labels are not yet finalized, the same pipelines work for semi/weak supervision and for post-hoc scoring against heuristics.

3) Method A — Treat Separate TCID (Row-Level Modeling)

Why this approach

- Maximizes number of training examples.
- Captures *within-session* behavioral shifts across time or tabs.
- Suitable for near-real-time scoring as new rows arrive.

Feature set (row-level)

- All engineered features above computed per row.
- Optional short-horizon aggregations within (tcid, tbid) windows (e.g., last 5 minutes).

Models evaluated

- Baseline: Logistic Regression (L2, class weighting), Linear SVM.
- Tree-based: Random Forest, Gradient Boosted Trees (XGBoost/LightGBM/CatBoost).
- Calibration: Platt scaling/Isotonic on validation folds for stable probabilities.

Evaluation design

- Leakage control: Grouped splits by tcid (no tcid appears in both train/val).
- Temporal realism: Optional time-based split with validation on later timestamps.
- Metrics: PR-AUC (positive minority focus), ROC-AUC, Recall@fixed FPR (e.g., 1%), Precision@Top-K, confusion matrix.
- Class imbalance: class weights or focal loss; threshold tuning vs. business cost.

Findings (qualitative)

- Zero/near-zero engagement rows and anomalous scroll-depth patterns are strong signals.
- Depth-per-scroll and events-per-second contribute disproportionately in tree models.
- User-agent families cluster distinctively; certain device types correlate with automation.

Risks & mitigations

- Label leakage: repeated tcid rows can inflate metrics → use GroupKFold and report PR-AUC on grouped CV.
- Non-stationarity: re-train periodically; apply drift monitors on marginal feature distributions.
- Outliers: clip/winsorize, or use robust models (tree-based) + anomaly flags.

4) Method B — Grouped TCID (Session-Level Modeling)

Why this approach

- Reduces noise from multiple rows per session; aligns with many anti-abuse actions made at the session/account level.
- Naturally avoids per-tcid leakage across folds; yields cleaner generalization estimates.

Aggregations (per tcid)

- Central tendency: mean/median of counts and depths.
- Extremes: max/min; 90th/95th percentiles of rate features.
- Coverage: share of zero-activity rows; share of anomaly-flagged rows.
- Stability: variance/entropy of interactions across the tcid timeline.
- UA mix: most frequent user_agent family; diversity of UA families.

Models evaluated

- Same family as Method A; tree ensembles typically excel on mixed numeric/categorical aggregates.
- Optional stacked model: linear model on top of tree logits for calibration.

Evaluation design

- Group-aware CV: GroupKFold by tcid for hyperparameter selection.
- Holdout: Optional final evaluation on the most recent tcids.
- Metrics: PR-AUC / ROC-AUC / Recall@FPR, plus lift at top deciles for ops triage.

Findings (qualitative)

- Aggregates smooth row-level noise; precision at low FPR generally improves.
- High shares of zero-scroll or zero-visibility rows and extreme depth ratios remain top features.
- tbid frequency (same tab observed many times) is a useful botness prior.

Trade-offs

- Slightly fewer training examples (one per tcid) and latency (must wait to aggregate).
- In exchange, cleaner labels and better stability for account/session-level decisions.

5) Comparative Summary (A vs B)

Aspect	Method A: Separate TCID	Method B: Grouped TCID
Granularity	Row-level (fine, real-time)	Session-level (coarse, decisive)
Leakage risk	Higher (repeated tcid)	Lower (one row per tcid)
Noise	Higher; sensitive to outliers	Lower; aggregates damp spikes

Latency	Low (streaming-friendly)	Medium (need aggregation window)
Typical use	Inline scoring, early warning	Final decisioning/triage

Recommendation: run a hybrid:

- Real-time row model for immediate risk scoring and rate-limits.
- Daily session model for durable actions and analyst review queues.
- Share a common feature library to keep signals consistent.