

A diagram of a flowchart

AI-generated content may be incorrect.

IMPLEMENTATION WORKFLOW

* **Bridge**: link each **ServiceNow incident** to the **most relevant device/app/day** slice with a **calibrated match score**.
* **Insights**: turn the bridged table into KPIs (MTTA, MTTR, reopen rate, SLA risk) and diagnostic views (device/app health vs incidents).

**DATA PRE-PROCESSING**

**1) Pre-merge data (staging & modeling)**

**1.1 Source → Staging (idempotent)**

* Normalize timestamps to **UTC**; keep source\_tz.
* Lowercase & trim emails; strip +aliases.
* Create **surrogate keys** and **hash PII** in analytics layers.

**1.2 Minimal schemas (star-ish)**

**dim\_employee**(employee\_sk, email\_hash, email\_raw\_sec, dept, title, location, hire\_date, manager\_email\_hash, …)  
**dim\_device**(device\_sk, device\_name\_norm, device\_aliases[], serial\_hash, os\_version, cpu\_model, is\_virtual, site, country, …)  
**dim\_application**(app\_sk, app\_name, vendor)  
**dim\_date**(date\_sk, calendar\_date, week, month, dow, …)

**fct\_incident**(incident\_sk, incident\_number, sys\_id, opened\_at\_utc, closed\_at\_utc, priority, severity, urgency, category, subcategory, assignment\_group, opened\_for\_employee\_sk, location, time\_worked\_mins, reopened\_count, …)

**fct\_device\_day**(device\_sk, date\_sk, health\_score, boot\_score, cpu\_util\_avg, mem\_util\_avg, disk\_q\_max, bitlocker\_status, …)

**fct\_device\_app\_day**(device\_sk, app\_sk, date\_sk, app\_usage\_s, app\_crashes, activity\_backend\_ms, remote\_display\_latency\_ms, …)

Explode/flatten Aternity’s \*\_set arrays here.

**alias maps**: device\_alias\_map(source\_name, device\_name\_norm, rule), email\_alias\_map(source\_email, canonical\_email)

**2) Candidate generation (blocking)**

Goal: reduce comparisons from millions to **dozens per incident**.

**Windows (defaults)**

* **pre**: −48h → open
* **during**: open → close
* **post**: close → +24h

**Primary blocks (any hit qualifies)**

1. opened\_for\_email == canonical\_email (exact, via alias map)
2. device\_name\_norm fuzzy match **within ±3 days** of incident open
3. Same employee\_sk & |incident\_open\_date − calendar\_date| ≤ 3

**Secondary filters**

* If incident text hints an app (from a small **curated dictionary**), restrict to that app\_sk.
* Keep device days with **activity** (e.g., app\_usage\_s>0 or health\_score present).

**ML PIPELINE**

**3) Feature building (no LLM needed)**

We compute features for each **(incident, candidate)** pair.

**3.1 Identity & string similarity**

* sim\_email\_exact ∈ {0,1}
* **Device name**:
  + jw\_device = Jaro–Winkler(name\_incident, name\_candidate)
  + jaccard\_3gram on character trigrams
  + soft\_tfidf (TF-IDF cosine over 3-grams with soft matching)
* **Auxiliary**: same\_department, same\_site, same\_country, is\_virtual\_match (XOR penalty)

**3.2 Temporal alignment**

* delta\_open\_vs\_day\_mins (0 means aligned)
* is\_pre / is\_during / is\_post
* days\_since\_last\_boot, time\_to\_close\_mins buckets

**3.3 Device posture & anomalies (z-scores)**

* cpu\_util\_avg\_z3d, mem\_util\_avg\_z3d, disk\_q\_max\_z3d  
  (candidate value minus rolling 3-day mean divided by std)
* health\_score\_delta\_3d (today − mean of previous 3 days)

**3.4 App usage & performance**

* app\_usage\_s, app\_crashes, remote\_display\_latency\_ms, activity\_backend\_ms
* app\_latency\_z3d, crash\_rate\_per\_hr (normalize by usage)
* is\_app\_active\_flag (usage > threshold)

**3.5 Incident text (lightweight, optional)**

* Keyword detectors (no LLM): binary flags for {teams, outlook, vpn, citrix, printer, zoom} found in short\_description|work\_notes|category|subcategory.
* keyword\_app\_match ∈ {0,1} (dictionary map → app\_sk)

**3.6 Quality & missingness**

* missing\_device\_info\_ratio, missing\_app\_metrics\_ratio (penalize)

**All features are numeric/categorical and stable.** Store them as a features\_json alongside each predicted link for audit.

**4) Similarity → Match score**

**4.1 Baseline composite (for transparent fallback)**

If you need a deterministic score before ML ships:

S = 0.55\*sim\_email\_exact

+ 0.20\*max(jw\_device, soft\_tfidf)

+ 0.10\*(1 - min(|delta\_open\_vs\_day\_mins|, 1440)/1440)

+ 0.05\*keyword\_app\_match

+ 0.05\*clip(health\_score\_delta\_3d\_pos, 0, 1)

+ 0.05\*is\_app\_active\_flag

* Accept Top-1 if **S ≥ 0.80**; **0.60–0.80** → low-confidence queue.

**4.2 ML scoring (recommended)**

Use **LightGBM** binary classifier on the full feature set; output **p(match)**.

* Loss: binary\_logloss with class weights (pos\_weight≈10–30)
* CV: **time-based splits** (train on older weeks, test on newer)
* Calibration: **isotonic** (reliable p)

**Threshold selection**

* Choose τ to optimize **Fβ**, β<1 to favor **precision** (e.g., β=0.5) OR set τ so that **Precision@1 ≥ 0.97** while maximizing **Coverage**.

**Advanced (optional later)**

* **Siamese** char-encoder for device names + MLP for numerics; cosine similarity head; trained with **contrastive loss**. Gives robustness to noisy device strings.

**5) Training data creation**

**5.1 Positive pairs (weak labels)**

* (incident, candidate) where:
  + opened\_for\_email exact equals candidate email **AND**
  + |incident\_open\_date − candidate\_date| ≤ 1 **AND**
  + device name similarity ≥0.9 **OR** same device\_sk.  
    These are high-precision seed positives.

**5.2 Negatives (hard and realistic)**

* Same incident paired with:
  + Same email, **different device** same day (hard negative)
  + Same device, **different email/user** same day (shared device case)
  + Nearby days with no activity in relevant app
* Ratio: start **1:5 (pos:neg)**, tune via AUC/PR.

**5.3 Active learning loop**

* Sample **low-confidence** predictions (0.6–0.8) to label 50–200 pairs/week.
* Retrain monthly; re-calibrate.

**6) Edge cases & rules (baked into features/decisioning)**

1. **Shared/pooled devices** (call centers/VDI):
   * Penalize when same device but email mismatch and is\_virtual=True.
   * Require either keyword\_app\_match=1 or strong temporal proximity to accept.
2. **Renamed devices / aliasing**:
   * Maintain device\_alias\_map (regex & learned pairs).
   * Give **extra weight** to historical co-occurrence (email, device\_name).
3. **No email on incident** (created by desk):
   * Rely on assignment\_group + location + keyword\_app\_match + temporal proximity + device activity.
4. **Multiple incidents same day**:
   * Allow **one device day** to link to multiple incidents; evidence JSON preserves traceability.
5. **Missing telemetry** (device offline):
   * If missing\_app\_metrics\_ratio>0.8, down-weight score; only accept with very strong identity/temporal match.
6. **Time zone drift**:
   * All to UTC; include tz\_offset\_minutes feature (helps model learn systematic shifts).
7. **Duplicates/reopens**:
   * Use (incident\_number, sys\_id) uniqueness; for reopens, keep separate rows but add a is\_reopen flag to features.

**7) Evaluation framework**

**Pairwise/Ranking**

* **PR-AUC**, **ROC-AUC**, **Precision@1**, **Recall@τ**, **MRR@10**, **Hits@1**
* **Coverage** = % incidents with ≥1 candidate above τ
* **Calibration**: Brier score, reliability curves

**Business KPIs (post-bridge)**

* Delta in **MTTR** explained by device/app metrics (R² of regression)
* **Reopen Rate** lift for incidents with poor device/app signals
* **Assignment accuracy** improvement in pilot (if used to route)

**Slices (must-have)**

* By **assignment\_group**, **location**, **priority**, **app** (e.g., Teams/VPN), **is\_virtual**

**1) Canonicalize first (don’t let taxonomy noise leak into ML)**

**Goal:** one stable “language” across ServiceNow and Aternity/Bridge.

* **Crosswalk table** taxonomy\_map(system, raw\_category, raw\_subcategory, raw\_assignment\_group) → canonical\_{l1,l2,l3}
  + Build with rules + data: exact rules, synonym lists, and **fuzzy merges** (Jaro–Winkler ≥0.92 on normalized strings).
  + **Hierarchy backoff:** if l2/l3 unknown, map to parent (canonical\_l1="Endpoint", l2="OTHER").
  + Store **versioned**; any “unknown” stays explicit: canonical="UNKNOWN" (never silently dropped).
* **Weekly propose-merge job:** TF-IDF on category strings + mini-batch k-means to surface candidates for human approval. 10 minutes/week of curation keeps drift under control.

Opinionated: Don’t train on raw labels. Train on **canonical** + raw encodings (see §2) for resilience.

**2) Feature encoders built for unknowns & drift**

We treat categories/assignment groups as *features* (the target for the matcher is still binary “is this the right device/app/day?”).

**For every high-card categorical (category, subcategory, assignment\_group, site, device\_model, app\_name):**

1. **Frequency / count encoding**
   * freq\_x = log1p(count(x)) — always defined, good signal, no leakage.
2. **Target encoding (TE) with OOF & Bayesian shrinkage**
   * OOF (K-fold) to avoid leakage.
   * Smoothed:  
     [  
     \text{TE}(x)=\frac{\sum y + \alpha \cdot \mu}{n + \alpha}  
     ]  
     with (\mu)=global mean, (\alpha) tuned.
   * **Unseen at inference** → n=0 ⇒ TE = global mean (safe default).
   * **Hierarchical backoff:** if category:l2 unseen, backoff to category:l1, then global.
3. **Hashing trick (one-hot via feature hashing)**
   * 2¹⁴–2¹⁶ dims (e.g., 16384).
   * Unseen → **auto-mapped** to a bucket; no retrain needed for vocabulary growth.
   * Works great with LightGBM on sparse CSR.
4. **CatBoost encoding (optional)**
   * If you prefer boosting-native encoding, CatBoost’s ordered target stats give similar robustness.
5. **Char-CNN/Sentence SVD (optional deep-lite)**
   * For raw strings (device names, assignment group text), a tiny **char-CNN→avg-pool→32d** embedding or **TF-IDF→TruncatedSVD(64)** gives generalization to typos/never-seen tokens.

We keep **both canonical** and **raw encodings**. Canonical gives stability; raw encoders catch new variants fast.

**3) LightGBM handling of unknown categories**

* If you pass **pandas “category” dtype**, unseen categories at inference are treated as **missing** → go **default split direction**. We **explicitly add**:
  + an is\_unknown\_\* flag, and
  + a **hashed feature** for that field,  
    so the model has signal instead of only “missing”.
* Net effect: **no crash**, **predictable behavior**, and unknowns get reasonable, data-driven scores.

**4) Similarity features & final match score (precision-first)**

We aggressively anchor the score on **identity/time/usage** (which generalize), letting categories be *supporting* signals.

**Key similarity features** (all numeric):

* **Identity**
  + sim\_email\_exact ∈ {0,1}
  + jw\_device, jaccard\_3gram\_device, device\_seen\_with\_user\_past30d
* **Temporal**
  + |Δ(open\_time, candidate\_day)| (mins), is\_pre/is\_during/is\_post
* **Usage/Health**
  + app\_usage\_s, app\_crashes, latency\_z3d, health\_score\_delta\_3d
* **Category encodings**
  + Canonical L1/L2 TE (OOF), freq enc, hash vector
  + Assignment group TE + hash
* **Text-lite**
  + keyword\_app\_match (tiny dictionary)

**Scoring**

* **Model:** LightGBM (binary), class\_weight to fight imbalance.
* **Calibration:** Isotonic over validation → p\_match.
* **Decision bands:**
  + p ≥ 0.90 → **ACCEPT (Top-1)**
  + 0.70 ≤ p < 0.90 → **REVIEW** (active-learning pool)
  + < 0.70 → **REJECT**

Strong stance: we optimize **Precision@1 ≥ 0.97** first. Coverage grows via active learning.

**5) Edge cases (hard-coded heuristics + features)**

1. **Shared/VDI devices**
   * Feature: is\_virtual, users\_on\_device\_7d, concurrent\_users\_flag
   * Rule: require either **strong time proximity** or **app keyword match**; otherwise down-weight.
2. **No user on incident**
   * Backoff on assignment\_group, site, and app signals; increase threshold to 0.93 for acceptance.
3. **Renamed devices**
   * device\_alias\_map (regex + learned pairs from prior links), plus device\_seen\_with\_user\_past30d feature.
4. **Missing telemetry**
   * missing\_ratio > 0.8 adds a penalty; only accept if **email+time** are very strong.
5. **Multiple incidents same day**
   * Allow many-to-one; we still produce **evidence JSON** per link.

**6) Training data that survives drift**

* **Positives (seed, high-precision):** email exact + |Δday| ≤ 1 + (device sim ≥0.9 **or** historical co-occurrence).
* **Negatives (hard):** same incident with wrong device/user, nearby days, or inactive app.
* **Imbalance:** start 1:5 pos:neg; tune via PR-AUC.
* **Active learning:** label 100–200 “review band” pairs/week → retrain monthly → re-calibrate.

**7) Evaluation you’ll actually trust**

* **Pairwise/ranking:** PR-AUC, ROC-AUC, **Precision@1**, Recall@τ, MRR@10, Coverage.
* **Calibration:** Brier score + reliability curve.
* **Slices:** by canonical L1/L2, assignment group, site, VM flag.
* **Ablations:** drop category features → ensure score doesn’t collapse (it shouldn’t).

**8) Aggressive unknowns strategy (TL;DR)**

* **Before training:** force every raw label to one of {KNOWN, OTHER, UNKNOWN}; keep raw string encoders.
* **During training:** OOF target encodings + hashing + is\_unknown\_\*.
* **At inference:** unseen → TE backoff to prior, hashing provides signal, LightGBM treats category as missing but still uses hash+flags.
* **Over time:** weekly taxonomy merges + monthly retrain keep the model sharp.

**9) Implementation sketch (ready to build)**

# fit\_encoders.py

class TEEncoder:

def \_\_init\_\_(self, alpha=50, n\_splits=5, random\_state=42):

...

def fit\_transform(self, X, y, col): # OOF mean with smoothing

...

def transform(self, X, col, fallback\_prior=True):

...

def make\_cat\_features(df, y, cols\_hash, cols\_te):

feats = {}

for c in cols\_te:

feats[f"te\_{c}"] = TEEncoder(alpha=50).fit\_transform(df, y, c)

feats[f"freq\_{c}"] = np.log1p(df[c].map(df[c].value\_counts()).fillna(0))

feats[f"is\_unknown\_{c}"] = (df[c].isin(["UNKNOWN","OTHER"]) | df[c].isna()).astype(int)

X\_hash = FeatureHasher(n\_features=2\*\*15, input\_type='string').transform(

df[cols\_hash].astype(str).agg('|'.join, axis=1)

)

return feats, X\_hash # dense dict + sparse CSR

# train\_lgbm.py

lgbm = lgb.LGBMClassifier(

n\_estimators=1500, learning\_rate=0.03, num\_leaves=63,

min\_data\_in\_leaf=100, subsample=0.9, colsample\_bytree=0.8,

objective='binary', class\_weight={0:1, 1:10}, n\_jobs=-1

)

lgbm.fit(X\_train, y\_train, eval\_set=[(X\_val,y\_val)], verbose=200)

cal = IsotonicRegression(out\_of\_bounds="clip").fit(lgbm.predict\_proba(X\_val)[:,1], y\_val)

# inference.py

p\_raw = lgbm.predict\_proba(X)[:,1]

p = cal.predict(p\_raw)

decision = np.where(p >= 0.90, "ACCEPT", np.where(p >= 0.70, "REVIEW", "REJECT"))

**10) Where categories “matter” downstream (insights)**

* We keep both **canonical** and **raw-encoded** forms in the bridge output.
* Analytics uses **canonical** to avoid dashboard churn; the model still benefits from raw hashing/TE under the hood.

**Bottom line**

Yes—**this works, and it’s aggressive** against unknown/diverse classes:

* **Canonicalize** early,
* **Encode** with OOF target + hashing + unknown flags,
* **Anchor** match on identity/time/usage,
* **Calibrate** the score, and
* **Continuously merge & retrain**.

**1) Pre-merge data you’ll produce**

**Staging facts/dims** (as we aligned earlier) plus a temporary **candidate\_pairs** table:

* candidate\_pairs (pre-merge working table)
  + Keys: incident\_sk, device\_sk, date\_sk, app\_sk (nullable)
  + Raw features: email/device strings, times, site, dept, app metrics, device health
  + Derived features: (see §3)
  + **label\_soft** (float, 0..1; initially from weak supervision), **label\_source** (fs\_em, snorkel, pseudo, human)
  + **split\_tag** (time-based fold), **run\_id**

This table is the substrate for both unsupervised and supervised training.

**2) No-label bootstrapping (three tracks)**

Run these **in parallel**; ensemble them later for stability.

**Track A — Unsupervised Fellegi–Sunter (FS) with EM**

Treat field agreements as latent-variable record linkage:

* Agreements: email\_exact, device\_sim\_bin ∈ {high,med,low}, |Δtime| bucket, site\_match, keyword\_app\_match.
* EM learns m/u probabilities (match/non-match) **without labels**.
* Output: **LLR score** per pair and a calibrated **p\_fs**.

**Track B — Weak supervision (Snorkel-style)**

Write **labeling functions (LFs)** (high-precision rules), e.g.:

* LF\_email\_exact\_time≤1d ⇒ MATCH
* LF\_device\_sim≥0.92 & same\_user & time≤2d ⇒ MATCH
* LF\_diff\_email & same\_device & time≤1d ⇒ NONMATCH (hard negative)
* LF\_virtual & diff\_user & no\_app\_usage ⇒ NONMATCH
* Combine LFs via a **label model** to produce **probabilistic labels** p\_ws.
* Save label\_soft=p\_ws and label\_source='snorkel' in candidate\_pairs.

**Track C — Self/contrastive pretraining (strings only)**

Learn robust device/email string embeddings **without labels**:

* Positives: (device\_name of same user across adjacent weeks), (alias pairs from historical co-occurrence)
* Negatives: random users/devices, shuffled names
* Train **char-CNN/GRU Siamese** with triplet/InfoNCE loss → emb\_device(·)
* Use cosine similarity as a **feature**, not the final scorer.

Result: You now have **p\_fs** (FS), **p\_ws** (weak supervision), and **string embeddings** to enrich features.

**3) Feature building (unknown-friendly, taxonomy-agnostic)**

For each (incident, candidate):

**Identity & strings**

* sim\_email\_exact ∈ {0,1}
* jw\_device, jaccard3\_device, cosine(emb\_device\_inc, emb\_device\_cand)
* device\_seen\_with\_user\_30d (co-occurrence count)

**Temporal**

* delta\_open\_vs\_day\_mins, flags is\_pre/is\_during/is\_post

**Aternity usage/health**

* app\_usage\_s, app\_crashes, remote\_display\_latency\_ms,
* z-scores: cpu\_util\_avg\_z3d, latency\_z3d, health\_score\_delta\_3d

**Categoricals (diverse/unknown-proof)**

* **Hashing trick** on raw category/subcategory/assignment\_group/site/app\_name → 2¹⁵ buckets (works for unseen).
* **OOF Target Encoding** (smoothed) for **canonicalized** L1/L2 if/when you adopt a crosswalk; **fallback to global mean** for unseen.
* is\_unknown\_\* flags when values are NULL/OTHER/UNKNOWN.

**Weak-signals**

* keyword\_app\_match from a tiny dictionary (Teams/VPN/Outlook/Zoom/Citrix/Printer)

**Weak-supervision priors**

* p\_fs, p\_ws from §2 (as features only; final model still decides).

**4) Discriminative model & scoring**

**Model**: **LightGBM binary classifier** on the full feature set (dense + sparse hashed).  
**Training labels**: use label\_soft from Track-B; weight each pair by w = 2\*|label\_soft-0.5| (confident LFs weigh more).  
**Imbalance**: class\_weight {nonmatch:1, match:10}.  
**Splits**: strictly **time-based** (train on older weeks, validate on newer).  
**Calibration**: **Isotonic** on the validation fold → reliable p\_match.

**Final match score**

* p\_match = Calibrate(LGBM(features, weights))
* **Decision bands** (precision-first):
  + p ≥ 0.90 → **ACCEPT (Top-1)**
  + 0.70 ≤ p < 0.90 → **REVIEW** (active-learning queue)
  + < 0.70 → **REJECT**

Optional: blend with FS by **stacking**: train LGMB on features + [p\_fs, p\_ws]; this usually boosts stability when taxonomies drift.

**5) “No mapping” handling for categories (aggressive)**

You said there’s **no mapping** between ServiceNow and the Bridge/DEX taxonomy. That’s fine.

* The model **does not require** a crosswalk to start; it uses:
  + **Hashing** (vocabulary-free) to encode raw category/group strings;
  + **Unknown flags** for missing/novel labels;
  + **Identity/temporal/usage** features (which dominate).
* If/when you want cleaner analytics, add a **canonicalization crosswalk** later; the model already has slots for canonical TE but safely falls back to the **global prior** when unseen.

**6) Negatives, hard negatives, and PU learning**

Expect positive scarcity. We’ll **inflate** the negative side **without poisoning**:

* **Hard negatives**: same incident + (same user **but** different device) same day; or same device **but** different user; or time-far pairs.
* **PU variant (optional)**: estimate class prior (Elkan–Noto) on weak-labels; adjust decision threshold to keep **Precision@1 ≥ 0.97** even if some positives are unlabeled.

**7) Active learning loop (small, powerful)**

Every run, take the **review band (0.70–0.90)**:

* Sample 100–200 pairs/week across slices (site, VM, category).
* Human label → write back to candidate\_pairs with label\_source='human'.
* Retrain monthly; **re-calibrate**; thresholds stay stable.

**8) Edge cases (encoded + guarded)**

* **Shared/VDI devices**: features is\_virtual, users\_on\_device\_7d, concurrent\_users\_flag; rule: require stronger temporal alignment or app keyword; else keep in review band.
* **No user on incident**: boost reliance on site/assignment\_group/app metrics; raise accept threshold to 0.93 for this subset.
* **Missing telemetry**: missing\_ratio>0.8 adds penalty; accept only if identity+time are strong.
* **Renamed devices**: learn alias pairs from historic co-occurrence; feed into Siamese positives; keep device\_alias\_map to hard-boost.

**9) Evaluation (with no gold labels… yet)**

You still get trustworthy signals:

* **FS internal**: posterior calibration curves; convergence checks.
* **Discriminative**: PR-AUC, ROC-AUC, Precision@1, Coverage on **weak-label validation** (time-based).
* **Two-view agreement**: Agreement between FS and LGBM ≥ X% on high scores.
* **Human spot-checks**: 300–500 pairs as a tiny golden set → report true **Prec@1/Coverage**; iterate thresholds.
* **Business proxies** post-merge: bridged links should **explain variance** in MTTR/Reopen (increases trust).

**10) Deliverables you can run**

* dbt models for staging + candidate\_pairs materialization.
* Python package:
  + blocking.py, features.py, unsup\_fs.py (EM), snorkel\_label.py,  
    string\_embed.py, train\_lgbm.py, calibrate.py, inference.py
* CLI:
  + bridge bootstrap --since 30d (build candidates, FS EM, LFs → label\_soft)
  + bridge train --fold weekly (train + calibrate)
  + bridge score --threshold 0.90 (write bridge table + evidence JSON)
  + bridge review-sample --n 200 (dump review band for labeling)

**Bottom line (my take)**

* **You don’t need mappings or gold labels to start.**
* Use **FS-EM + weak supervision** to get **probabilistic “silver” labels**,
* Train **LightGBM** with **unknown-tolerant encodings** and **calibration**,
* Gate with **precision-first thresholds**, and
* Tighten via **small, targeted human labels** over time.

**FLOW UNDERSTANDING**

1. **Stage & normalize (idempotent)**
   * UTC timestamps; lowercase emails; strip +aliases.
   * Facts: fct\_incident, fct\_device\_day, fct\_device\_app\_day (explode Aternity sets).
   * Dims & maps: dim\_employee, dim\_device, dim\_application, device\_alias\_map, email\_alias\_map.
2. **Candidate generation (blocking)**
   * Blocks (any hit): email exact; fuzzy device name ±3 days; same employee ±3 days.
   * Filters: require activity/health on candidate day; restrict by app keyword if present.
3. **Unsupervised link evidence (no labels)**
   * Fellegi–Sunter with EM on agreement bins (email, device-sim bands, time buckets, site, keyword).
   * Output: p\_fs (posterior match probability per pair).
4. **Weak supervision (Snorkel-style)**
   * High-precision labeling functions (e.g., email+Δtime≤1d ⇒ match; diff-email+same-device ⇒ nonmatch).
   * Label model → p\_ws (probabilistic label).
   * Write candidate\_pairs with label\_soft=p\_ws, label\_source.
5. **Self/contrastive pre-training for strings**
   * Siamese char-CNN/GRU on device names using co-occurrence positives; produce embeddings.
   * Use cosine(emb\_inc, emb\_cand) as a feature (not classifier).
6. **Feature building (unknown-proof)**
   * Identity: email exact; Jaro-Winkler & 3-gram Jaccard; device-seen-with-user-30d.
   * Temporal: Δ(open vs day), flags pre/during/post.
   * Usage/health: app\_usage\_s, crashes, latency\_z3d, health\_delta\_3d.
   * Categoricals (diverse taxonomies):
     + Hashing trick on raw category/subcategory/group/site/app.
     + OOF target encoding on optional canonical L1/L2 (fallback to global mean).
     + is\_unknown\_\* flags.
   * Priors: include p\_fs, p\_ws as inputs (stacking).
7. **Discriminative model & calibration**
   * LightGBM (binary) with class imbalance handling; train on features with sample weight = 2\*|label\_soft−0.5|.
   * Time-based CV; Isotonic calibration on validation → reliable p\_match.
8. **Decision bands (business-safe)**
   * p\_match ≥ 0.90 → ACCEPT Top-1 and write bridge\_incident\_device\_app.
   * 0.70 ≤ p\_match < 0.90 → REVIEW (active learning queue).
   * < 0.70 → REJECT.
   * Persist features\_json (“why it matched”) with each ACCEPT.
9. **Active learning loop**
   * Weekly sample from REVIEW band (stratified by site/VM/category), human-label 100–200 pairs.
   * Append to candidate\_pairs; monthly retrain & recalibrate.
10. **Edge cases baked in**

* Shared/VDI: require stronger time/app evidence or keep in REVIEW.
* No user on incident: boost site/group/app signals; raise accept threshold to 0.93.
* Renamed devices: alias map + embedding similarity + historical co-occurrence.
* Missing telemetry: penalize; accept only with strong identity+time.

1. **Evaluation you can trust**

* PR-AUC, ROC-AUC, Precision@1, Coverage, MRR@10, Brier score (calibration).
* Slices by VM/site/group/category.
* Small golden set (300–500) to verify true Prec@1.