Customer Churn Early-Warning System

Executive Summary

Customer churn is rarely a surprise; it is usually preceded by visible signals such as reduced product usage, increasing support friction, plan downgrades, and explicit renewal concerns. This project delivers a practical early-warning dashboard that brings those signals into one place so a Customer Success team can act before a customer cancels. The dashboard produces a prioritized list of at-risk customers, explains why each customer is considered risky, and suggests what to do next, such as scheduling a check-in, providing onboarding support, or conducting a pricing/value review.

To strengthen insight from customer conversations, the system can also run a local language model (LLM) on support-ticket text to detect cancellation intent and other warning signals. Because the model runs locally, ticket data does not need to be sent to the internet.

Project Overview

This project is an interactive dashboard that helps a Customer Success team identify customers who may stop using the product or cancel soon. It combines everyday business signals—subscriptions, revenue, product usage, support activity, and churn events—into a single operational view. The dashboard outputs an early-warning action list showing which customers to contact first, why they appear risky, and what actions are recommended.

Problem Statement and Goal

Customer Success teams often learn about churn risk too late—after renewal discussions fail or after usage declines to near zero. The goal of this system is to provide a consistent, explainable, and operational method to detect churn risk early and convert insights into concrete next steps. This supports two outcomes: proactive retention outreach to the right customers, and improved understanding of what drives churn risk across customer segments such as plan tiers.

Datasets Used

The system uses two categories of data: structured SaaS data and unstructured ticket text.

Structured SaaS Data (five CSV tables)

The SaaS dataset is split into five CSV files because each file represents a different business domain that typically comes from a different system (billing, product analytics, support operations, and churn tracking). Keeping them separate mirrors real-world pipelines and allows each domain to be refreshed independently while still being combined during analysis.

Accounts (accounts CSV)
This table describes who the customer is at the account level. It includes identifiers (`account_id`), descriptive metadata (`account_name`, `industry`, `country`), and lifecycle fields such as `signup_date` and `referral_source`. Fields like `plan_tier`, `seats`, `is_trial`, and `churn_flag` support segmentation and context.

Subscriptions (subscriptions CSV)
This table describes what the customer purchased and how billing works. Each subscription links back to an `account_id` and includes `start_date`, `end_date`, `plan_tier`, `billing_frequency`, `auto_renew_flag`, and revenue fields such as `mrr_amount` and `arr_amount`. Upgrade and downgrade flags are important because plan changes often precede churn.

Usage (feature usage CSV)
This table captures product usage over time at the subscription level. It typically includes `usage_date`, `feature_name`, `usage_count`, and `error_count`. This table enables trend signals such as usage drop in the most recent 30 days compared to the prior 30 days.

Churn Events (churn CSV)
This table records churn outcomes and churn explanations. Each churn event links to `account_id` and includes `churn_date`, `reason_code`, and additional context such as `refund_amount_usd` or preceding plan change flags. This table provides the churn label and supports churn-reason reporting.

Support Tickets (support tickets CSV)
This table captures customer friction and support experience. It links to `account_id` and includes `submitted_at`, `resolution_time_hours`, `first_response_time_minutes`, `satisfaction_score`, and `escalation_flag`. Ticket volume and poor support experience often correlate with churn risk.

Unstructured Ticket Dataset (ticket text CSV, optional)

A separate support-ticket text dataset can be used for LLM analysis and demos. It typically includes free-text fields such as `subject` and `body` (and sometimes tags, queue, priority, language, and version). If a true `customer_id` is not present to map tickets to the SaaS customers, the ticket dataset is treated as a demo dataset for showcasing text extraction, not as a production-grade customer linkage.

Architecture

The system architecture consists of several key components.

Data Sources
Structured SaaS data (accounts, subscriptions, usage, churn events, support tickets) and optional unstructured ticket text.

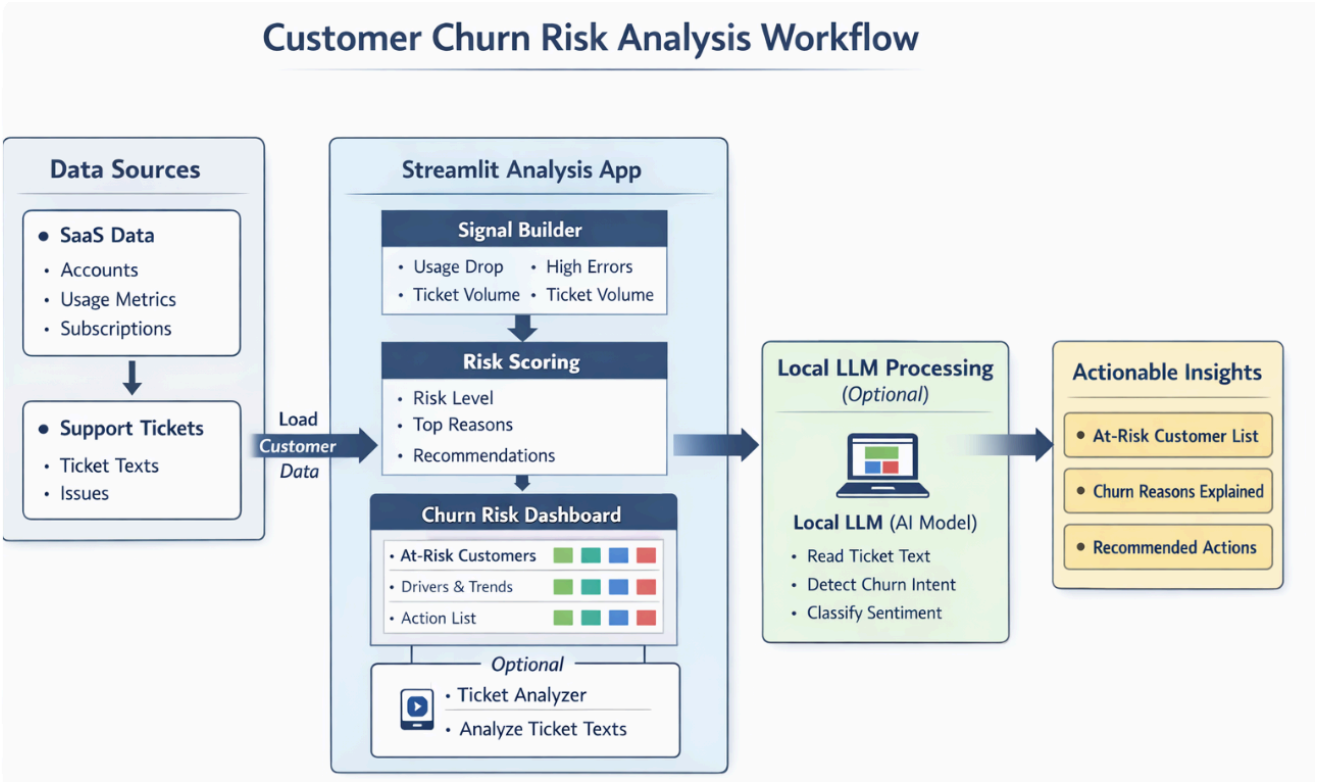
Processing Pipeline
Data loading, datatype normalization (dates and IDs), customer health metric computation, and optional LLM ticket enrichment.

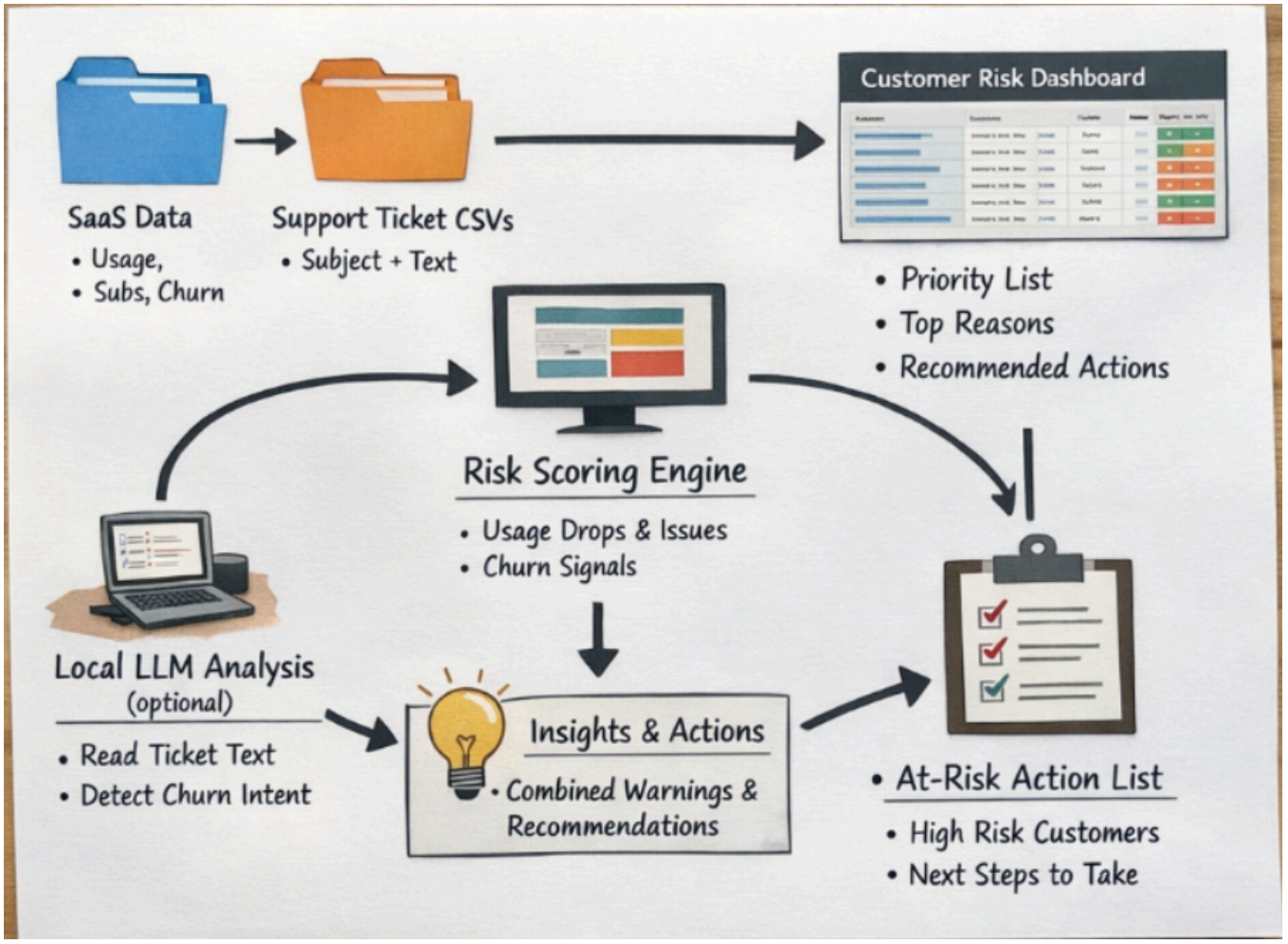
Scoring Engine
A rule-based scoring layer converts computed signals (usage drop, support load, tenure/onboarding risk, and optional churn intent from ticket text) into a risk score, a risk level, and an explanation. The output includes "top reasons" and "recommended actions" so that results are explainable and operational.

Dashboard UI
Streamlit is used to provide an interactive dashboard with four tabs: Executive, Drivers, At-Risk List, and Ticket Analyzer.

LLM Runtime (local)
A locally installed Ollama model (for example, llama3.1:8b or llama3.1:3b depending on performance needs) can enrich ticket text by producing structured JSON fields such as summary, category, sentiment, churn intent, and recommended action. This enrichment is optional because the dashboard can run fully on structured SaaS signals even without ticket text analysis.

This architecture enables integration of structured and unstructured data to provide a churn risk early-warning system with actionable insights. Figure 1 shows the end-to-end flow from raw CSVs to risk scoring and dashboard outputs.





Tools and Implementation

The system is implemented using Python. Streamlit provides the dashboard interface; Pandas and NumPy perform data loading and transformations; Ollama provides the local runtime for running the selected LLM; and CSV files act as the dataset layer for input data.

End-to-End Method (How the System Works)

The workflow begins by loading the SaaS CSV files and normalizing IDs and dates. Subscriptions are treated as "customers" at the scoring level (customer_id is derived from subscription_id), and account-level attributes (such as account_name and industry) are joined in to make the dashboard readable.

Next, customer health metrics are computed. Usage is aggregated into two windows: the most recent 30 days and the prior 30 days. This supports a directional signal—whether usage is trending down. Support tickets are aggregated for the last 30 days to reflect recent friction. Churn events are used to label accounts as churned (for evaluation and reporting).

Finally, the scoring engine evaluates each customer row and assigns a churn risk score (0–100), a risk level (High/Medium/Low), top reasons, and recommended actions. If the ticket LLM is enabled, ticket signals such as churn intent can be added as an extra input to scoring.

Risk Scoring Method (Explainable Rules)

The churn risk score is calculated using an explainable, rule-based method. Each customer begins with a score of 0. The function then adds points when specific risk signals are detected. These signals come from product usage trends, support activity, product errors, ticket sentiment signals (from the LLM), and account lifecycle indicators such as trial status or downgrades.

The final output contains four values:

1. **Risk score** (0–100)
2. **Risk level** (High / Medium / Low)
3. **Top reasons** (up to 3 key reasons that explain the score)
4. **Recommended actions** (up to 3 suggested actions based on the reasons)

The score is capped at 100, meaning even if multiple signals trigger, the maximum possible score remains 100.

Inputs used by the scoring function

The following fields are read from the customer record:

- **usage_drop_pct**: percentage drop in usage (last 30 days vs prior 30 days), stored as a decimal (0.60 = 60%)
- **tickets_30d**: number of support tickets in the last 30 days
- **errors_30d**: number of product errors in the last 30 days
- **churn_intent_30d**: count of LLM tickets flagged as churn intent in the last 30 days
- **satisfaction_score**: support satisfaction score (if available), typically 1–5
- **escalation_flag**: whether a ticket escalation occurred (1 = yes)
- **downgrade_flag**: whether the customer downgraded recently (1 = yes)
- **is_trial**: whether the customer is a trial customer (True/False)

Step-by-step scoring rules (how points are added)

1) Usage drop (core churn signal)

Usage drop is treated as the most important signal and contributes the largest part of the score.

- If **usage_drop_pct ≥ 0.60** (60%+ drop) → **+40 points**
- Else if **usage_drop_pct ≥ 0.40** (40%+ drop) → **+28 points**
- Else if **usage_drop_pct ≥ 0.20** (20%+ drop) → **+12 points**
- Otherwise → **+0 points**

A reason is added, such as:
“Usage dropped 75% (30d vs prior)”

2) Support load (ticket volume)

A high number of tickets is treated as a churn risk signal because it often reflects friction or dissatisfaction.

- If **tickets_30d ≥ 6 → +18 points**
- Else if **tickets_30d ≥ 3 → +10 points**
- Otherwise → **+0 points**

A reason is added such as:
“High support load (6 tickets in 30d)”

3) Product quality/friction (errors)

Errors represent a product-quality issue that can increase churn risk.

- If **errors_30d ≥ 10 → +15 points**
- Else if **errors_30d ≥ 3 → +8 points**
- Otherwise → **+0 points**

A reason is added such as:
“High error volume (12 errors in 30d)”

4) Explicit churn intent from LLM

If the LLM detects cancellation intent in ticket text, that is treated as a strong risk indicator.

- If **churn_intent_30d ≥ 1 → +22 points**

A reason is added:
“Customer language suggests cancellation risk”

This rule does not scale with the number of churn-intent tickets; it only checks whether it is at least one.

5) Support satisfaction score (if present)

If a satisfaction score exists and is valid, additional risk points are added.

- If **satisfaction_score ≤ 2 → +12 points**
  Reason: “Low satisfaction score (1–2/5)”
- Else if **satisfaction_score == 3 → +6 points**
  Reason: “Medium satisfaction score (3/5)”
- Otherwise (4–5) → **+0 points**

This block is wrapped in a safe conversion so missing/invalid values do not crash scoring.

6) Escalation/downgrade / trial risk

These are additional lifecycle risk factors.

Escalation

- If **escalation_flag == 1 → +10 points**
  Reason: “Ticket escalation (higher risk)”

Downgrade

- If **downgrade_flag == 1 → +12 points**
  Reason: “Recent downgrade (value risk)”

Trial

- If **is_trial == True → +8 points**
  Reason: “Trial customer (activation risk)”

Score cap and risk level thresholds

After adding all points, the score is capped:

- **risk_score = min(score, 100)**

The risk level is then assigned using fixed thresholds:

- **High** if **risk_score ≥ 60**
- **Medium** if **risk_score ≥ 30** and < 60
- **Low** if < 30

How “Top reasons” are chosen

All triggered reasons are collected in a list (in the same order as the scoring rules).
Only the first three reasons are shown in the dashboard:

- **top_reasons = first 3 reasons joined with “; ”**

This means the dashboard highlights the most important reasons first (usage drop and support load are evaluated early, so they typically appear first when present).

How “Recommended actions” are generated

Actions are not calculated mathematically. They are generated using rules that map reasons to recommended playbook steps.

The function checks which types of reasons were triggered and appends related actions. Only the first three actions are displayed:

- **recommended_actions = first 3 actions joined with “; ”**

Action rules:

If a usage drop reason exists

Actions added:

- “Schedule a check-in to reset goals”
- “Recommend 1–2 key features to activate this week”

If support load or escalation exists

Actions added:

- “Confirm resolution plan and close the loop”
- “Escalate repeat issues with a clear timeline”

If cancellation risk exists (LLM churn intent)

Actions added:

- “Offer a value review (ROI + quick wins)”
- “Create a 2-week success plan with milestones”

If trial risk exists

Actions added:

- “Offer guided onboarding session”
- “Share onboarding checklist + milestones”

If low satisfaction exists

Actions added:

- “Follow up on dissatisfaction with a recovery plan”
- “Offer proactive support for the next 7 days”

If downgrade exists

Actions added:

- “Run a value-gap call to understand missing needs”
- “Offer training or enablement for key workflows”

If no rule matches, a default action is returned:

- “Maintain regular touchpoints and monitor health signals”

Worked examples (how the numbers add up)

Example 1: High-risk customer (score ≥ 60)

Signals:

- usage_drop_pct = 0.60 → +40
- tickets_30d = 6 → +18
- churn_intent_30d = 1 → +22
  Total = 40 + 18 + 22 = 80 → **High**

Top reasons (first 3):

- Usage dropped 60% (30d vs prior)
- High support load (6 tickets in 30d)
- Customer language suggests cancellation risk

Recommended actions (first 3):

- Schedule a check-in to reset goals
- Recommend 1–2 key features to activate this week
- Confirm resolution plan and close the loop

Example 2: Medium-risk customer (30–59)

Signals:

- usage_drop_pct = 0.40 → +28
- errors_30d = 3 → +8
- is_trial = True → +8
  Total = 28 + 8 + 8 = 44 → **Medium**

Example 3: Low-risk customer (< 30)

Signals:

- tickets_30d = 3 → +10
  Total = 10 → **Low**

This scoring approach is deliberately rule-based. Each score can be traced back to specific triggered conditions, and each condition adds a known number of points. This makes the output easy to explain to stakeholders and easy to tune (adjusting thresholds and weights changes behavior in a controlled way)

Dashboard preview

The dashboard is organized into four tabs: Executive, Drivers, At-Risk List, and Ticket Analyzer. It also describes the user controls for selecting the number of tickets to process with the LLM and setting the timeout per ticket, explaining how these settings help optimize ticket analysis workload and processing time.
Ticket data can be loaded from:

- **Folder mode:** choose a CSV from `data/raw/tickets` using a dropdown, or
- **Upload mode:** upload a ticket CSV directly from the dashboard, which is then saved locally and becomes available for processing.

This design allows quick testing of different ticket datasets without editing the source code.

SaaS dataset upload
The SaaS data model currently expects five related tables because each table contributes different signals (accounts, subscriptions, usage, churn events, support tickets). A SaaS "Upload" mode would allow replacing these files through the UI (either as five separate uploads or a zipped bundle), enabling analysts to evaluate alternate datasets without code changes.

Timeout Behavior and What It Means

The dashboard includes an LLM timeout control ("Timeout per ticket (seconds)") to prevent the application from hanging while analyzing ticket text. When the LLM is slow or the system is CPU-constrained, an output may appear like:

```
"error": "timeout_after_11s"
```

This output means the model did not return a response within the selected time limit. It is not a crash; it is a **protective safeguard**. The fields such as summary/category/sentiment remain empty because the model did not finish processing in time. Increasing the timeout, reducing "Tickets to process," or using a smaller local model typically improves completion.

LLM

The system is designed so that the churn early-warning engine works even when the LLM is disabled or timeouts occur. The churn risk scoring is primarily built on **structured signals** (usage drops, support load, churn history, tenure, plan changes). The LLM adds an additional layer: it reads ticket text to detect **language signals** such as cancellation intent, frustration, pricing pushback, or urgency.

The LLM is described as optional for three reasons:

1. **Privacy and deployment:** ticket text can be sensitive; local processing avoids sending data to external services.

2. **Reliability:** scoring can still run using only structured signals if the model is slow, unavailable, or rate-limited.

3. **Cost and compute:** local LLM inference can be heavy on CPU; structured scoring remains fast and consistent.

With LLM, the dashboard performs **automated reasoning + decision support**, not just visualization. It detects risk, explains drivers, and generates actions and outreach drafts. The LLM simply enhances the agent with natural language understanding when available.

Local LLM Runtime

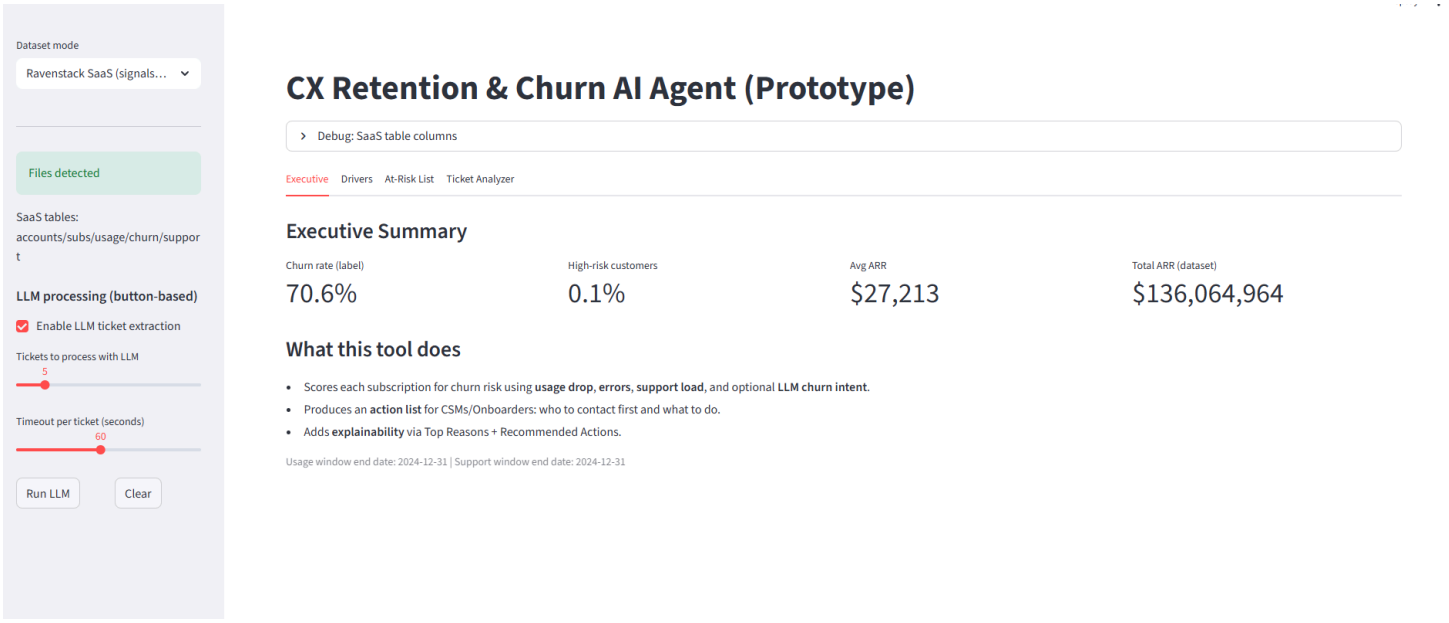
The project uses a locally installed Ollama model (for example `llama3.1:3b` or `llama3.1:8b`). This prototype was tested with llama3.1:3b for speed on CPU and llama3.1:8b for better quality when more compute is available. A local model was chosen because it is practical for demos, supports offline processing, and keeps sensitive ticket text on the same machine. The model's role is not to "predict churn" mathematically; it is to convert unstructured ticket text into structured fields such as:

- short summary of the issue,
- category (billing, bug, onboarding, performance, etc.),
- sentiment (negative/neutral/positive),
- churn intent flag,
- recommended action.

These fields can then be used as additional signals alongside usage/support metrics.

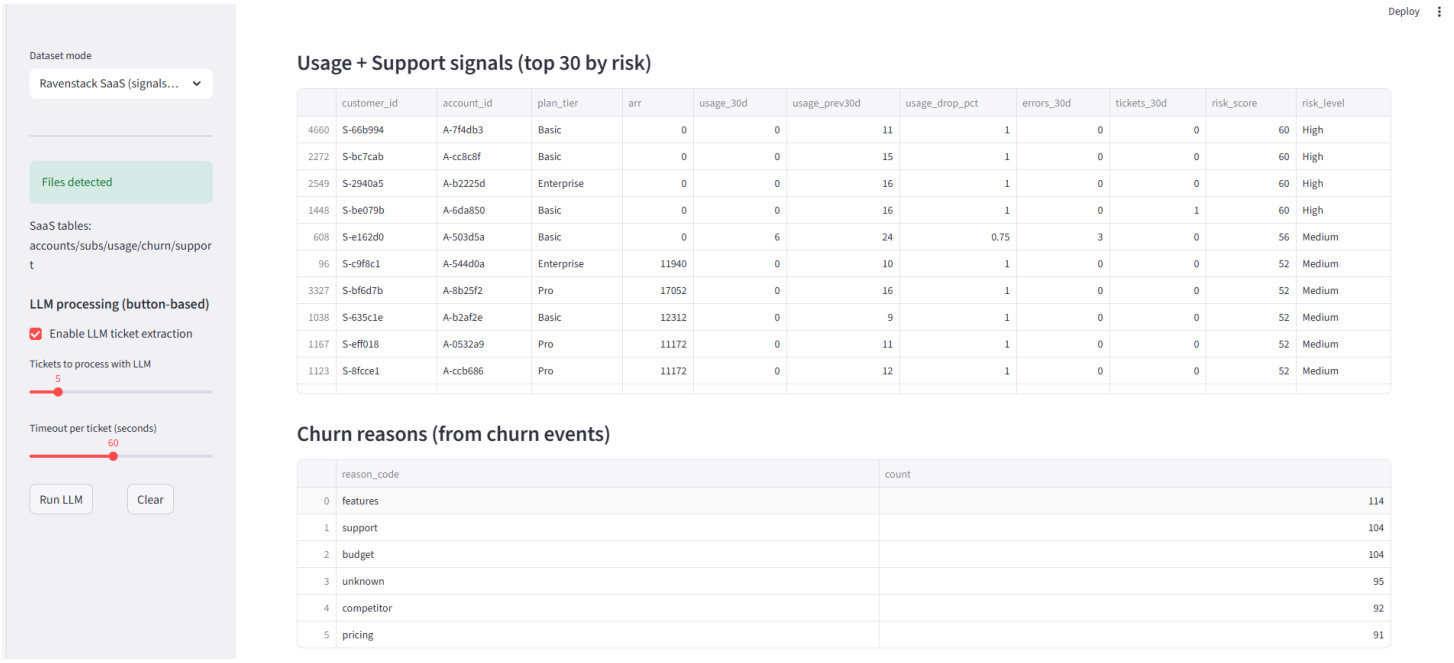
- **Executive Summary**

  The Executive tab is designed for leadership weekly review. It answers: How risky is the customer base overall, and how much revenue is potentially at risk? It summarizes churn rate (from churn labels), the percentage of customers classified as high risk, and revenue context such as average ARR and total ARR in the dataset. This tab helps stakeholders track overall retention health without reading individual customer details.



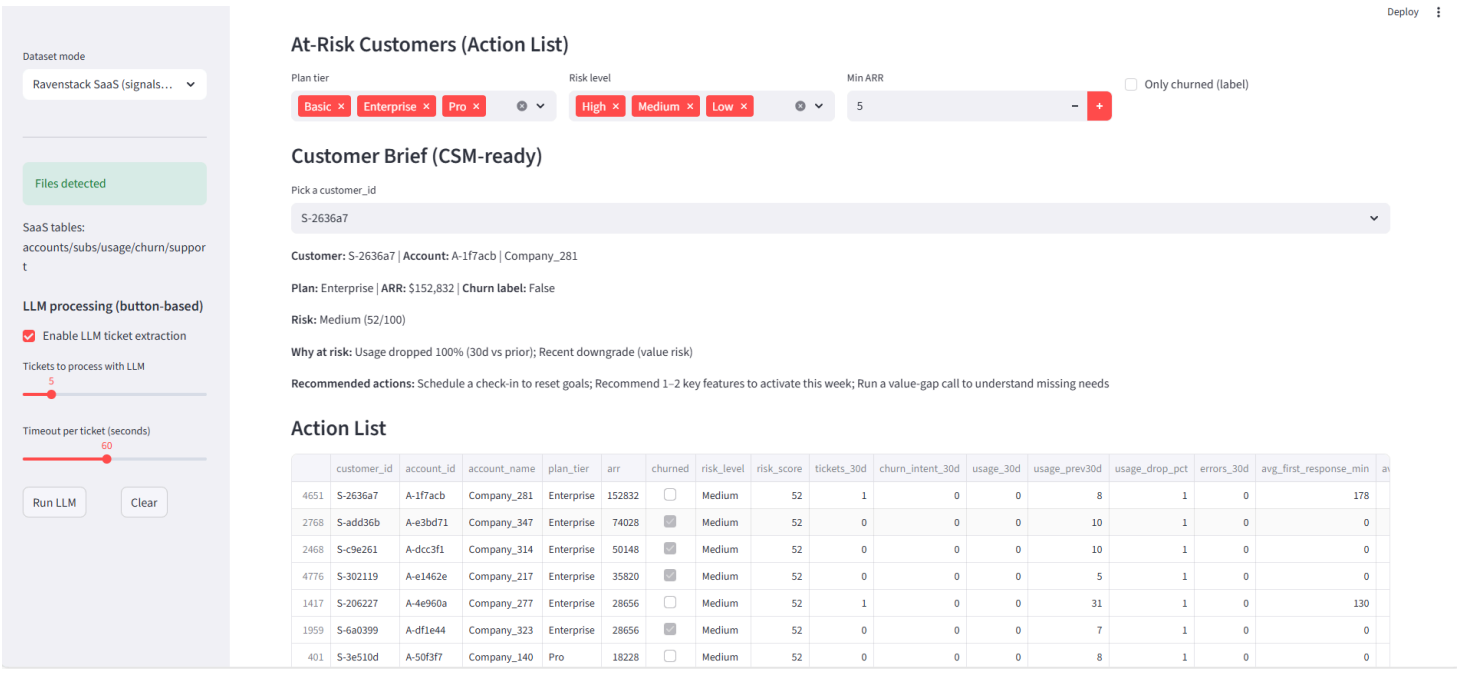
- **Drivers & Insights**

The Drivers tab is designed for data analysts and CX managers. It explains why customers are being flagged so teams can focus on the biggest systemic problems (for example onboarding gaps, support quality issues, or product adoption problems). The plan tier table shows how churn risk differs across plans (Basic vs Pro vs Enterprise), which helps answer whether churn risk is concentrated in a specific segment. The churn driver chart shows how often each risk reason appears across customers; the x-axis lists driver categories (such as usage drop or support load), and the y-axis shows the number of customers flagged for each driver. The churn reasons section summarizes churn event reason codes and how frequently each appears in churn events.



- **At-Risk List**

  The At-Risk tab is the operational queue for Customer Success execution. Filters narrow the list to the segment that matters (for example, only High risk or only customers above a minimum ARR). Selecting a customer shows a brief summary explaining why the customer is at risk and what to do next. This tab can also provide structured "agent-like" support such as a step-by-step next-best-action workflow, outreach drafts, and session memory notes to track follow-up outcomes. "Agent Actions" area appears after selecting a customer from the At-Risk List. It contains three practical features designed to help Customer Success teams act immediately.

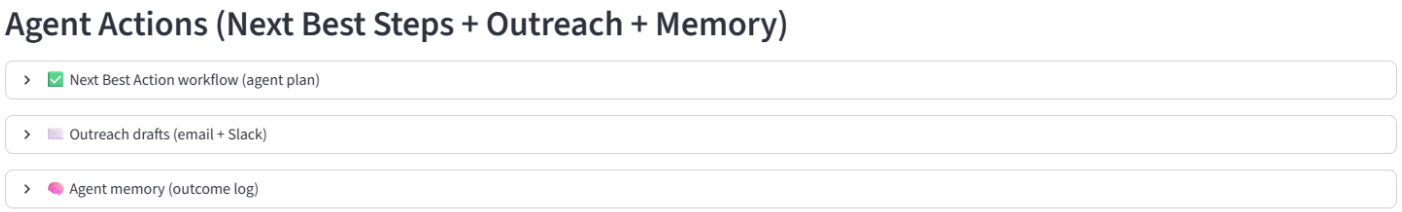


Next Best Action (workflow checklist).
This feature generates a step-by-step plan based on the customer's risk reasons (for example: usage dropped, support load is high, onboarding risk, or cancellation intent). The plan is rule-based and produces a structured checklist such as "schedule check-in," "confirm success criteria," "activate key features," and "follow up in 7

days.” This makes the system feel agent-like because it converts analysis into an executable workflow rather than leaving the user to interpret data manually.

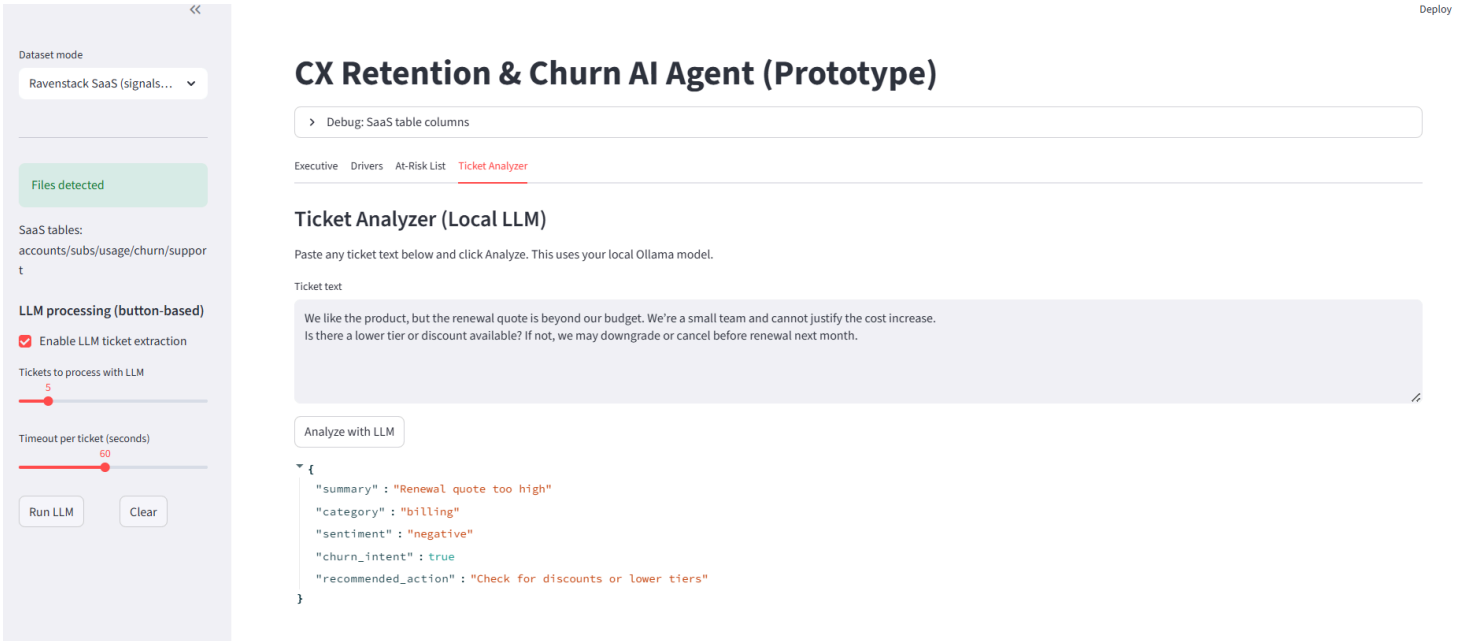
Outreach Drafts (Email + Slack templates).
This feature creates communication drafts automatically using the customer’s plan tier, ARR, top reasons, and recommended actions. It does not require the language model. The output includes an email subject, email body, and a short Slack message draft that can be copy-pasted into the team’s workflow. This removes friction for Customer Success teams by shortening the time from detection to outreach.

Agent Memory (session outcome log).
This feature stores a simple follow-up record inside the Streamlit session, including an outcome (contacted, meeting scheduled, issue resolved, renewed, downgraded, cancelled, no response) plus optional notes. This acts as an “agent memory” within the session so follow-ups are not lost during the demo or review. It creates a minimal feedback loop and makes the dashboard feel like an operational tool rather than a static report.



- **Ticket Analyzer**

The Ticket Analyzer tab runs the local LLM on a single pasted ticket and returns structured JSON. It helps validate how the model is interpreting customer language and demonstrates the text-to-signal step during presentations. For example, if a ticket says the renewal quote is too high and cancellation is being considered, the model can classify the ticket as pricing-related with churn intent and recommend a retention action such as a plan fit discussion or value review. If the model times out, the output will show a timeout error rather than freezing the app.



Features

- Combines three core capabilities:
    - Multi-source signal integration (usage, subscriptions, churn events, support tickets)
    - Explainable, rule-based risk scoring (clear reasons + recommended actions)
    - Execution-oriented dashboard (turns insights into follow-up steps)
- Enables interactive filtering and sorting in the **At-Risk List** to quickly focus on the right customer segment
- Supports near real-time refresh when the underlying datasets (CSVs) are updated
- Integrates structured metrics with **optional** unstructured ticket-text analysis (LLM-based insights)
- Produces churn-risk outputs that are easy to operationalize:
    - Risk score (0–100)
    - Risk level (High/Medium/Low)
    - Top reasons (why flagged)
    - Recommended actions (what to do next)
- Designed to support both audiences:
    - High-level stakeholder reporting (executive visibility)
    - Day-to-day Customer Success workflows (prioritized action queue + playbooks)

Outputs

The dashboard produces a churn risk score and risk level per customer, explanation text (“top reasons”), recommended actions, next-best-action workflow steps, outreach drafts (email and Slack templates), a session-level memory log for outcomes, and JSON output from the LLM.

Output: prioritized at-risk customer list with explanations and recommended actions.

Action List

	customer_id	account_id	account_name	plan_tier	arr	churned	risk_level	risk_score	tickets_30d	churn_intent_30d	usage_30d	usage_prev30d	usage_drop_pct	errors_30d	avg_first_response_min	a
4651	S-2636a7	A-1f7acb	Company_281	Enterprise	152832	☐	Medium	52	1	0	0	8	1	0	178	
2768	S-add36b	A-e3bd71	Company_347	Enterprise	74028	☑	Medium	52	0	0	0	10	1	0	0	
2468	S-c9e261	A-dcc3f1	Company_314	Enterprise	50148	☑	Medium	52	0	0	0	10	1	0	0	
4776	S-302119	A-e1462e	Company_217	Enterprise	35820	☑	Medium	52	0	0	0	5	1	0	0	
1417	S-206227	A-4e960a	Company_277	Enterprise	28656	☐	Medium	52	1	0	0	31	1	0	130	
1959	S-6a0399	A-df1e44	Company_323	Enterprise	28656	☑	Medium	52	0	0	0	7	1	0	0	
401	S-3e510d	A-50f3f7	Company_140	Pro	18228	☐	Medium	52	0	0	0	8	1	0	0	

Output: Top 30 customers at risk and reason of Churns

Usage + Support signals (top 30 by risk)

	customer_id	account_id	plan_tier	arr	usage_30d	usage_prev30d	usage_drop_pct	errors_30d	tickets_30d	risk_score	risk_level
4660	S-66b994	A-7f4db3	Basic	0	0	11	1	0	0	60	High
2272	S-bc7cab	A-cc8c8f	Basic	0	0	15	1	0	0	60	High
2549	S-2940a5	A-b2225d	Enterprise	0	0	16	1	0	0	60	High
1448	S-be079b	A-6da850	Basic	0	0	16	1	0	1	60	High
608	S-e162d0	A-503d5a	Basic	0	6	24	0.75	3	0	56	Medium
96	S-c9f8c1	A-544d0a	Enterprise	11940	0	10	1	0	0	52	Medium
3327	S-bf6d7b	A-8b25f2	Pro	17052	0	16	1	0	0	52	Medium
1038	S-635c1e	A-b2af2e	Basic	12312	0	9	1	0	0	52	Medium
1167	S-eff018	A-0532a9	Pro	11172	0	11	1	0	0	52	Medium
1123	S-8fcce1	A-ccb686	Pro	11172	0	12	1	0	0	52	Medium

Churn reasons (from churn events)

	reason_code	count
0	features	114
1	support	104
2	budget	104
3	unknown	95
4	competitor	92
5	pricing	91

Limitations

- **Rule-based scoring (not a trained predictive model)**
  - The churn “risk_score” is created using explainable rules, not ML training.
  - Because of that, it may **miss subtle churn patterns** and may not maximize accuracy metrics (precision/recall).
  - It is best for **early-warning + prioritization**, not “guaranteed churn prediction.”
- **Ticket-to-customer linkage may be demo-only**
  - If the external ticket dataset does **not** include a real `customer_id` / `account_id` that matches the SaaS dataset, the mapping becomes **synthetic** (random mapping in the prototype).
  - In that case, ticket insights are valid for **text understanding**, but not valid as a true customer-level churn feature.
- **Local LLM speed depends on the machine**
  - Local inference on CPU (especially for larger models) can be slow.
  - Low timeout values can produce outputs like `timeout_after_11s` even when the model is working correctly.
- **Data quality issues directly affect outputs**
  - Missing timestamps, inconsistent IDs, or incomplete usage/support records can reduce the reliability of calculated signals (usage trends, ticket volume, etc.).
- **Short analysis window**
  - Signals are currently computed mainly over recent windows (e.g., last 30 days vs prior 30 days).
  - Customers with seasonal usage patterns may be misclassified unless longer windows are added.

Challenges Encountered (What happened + how it was solved)

- **Challenge 1 — LLM batch processing getting "stuck."**
  - **What happened:** While processing multiple tickets (example: 5 tickets), the UI showed `LLM running 1/5 tickets processed` for a long time.
  - **Why it happened:** Local LLM inference can be slow on CPU, and one long ticket or slow generation can block the loop.
  - **How it was solved:**
    - Added a **hard timeout per ticket** using a background thread (`threading.Thread`) and `join(timeout=...)`.
    - Added a **Timeout per ticket (seconds)** slider so the system never freezes.
    - Reduced prompt size by trimming ticket input (e.g., limiting text length) to keep inference fast and predictable.
  - **Outcome:** The dashboard continues running even if the LLM fails or takes too long; each ticket either returns structured output or a controlled timeout error.
- **Challenge 2 — Ticket text size and "dirty" characters**
  - **What happened:** Some ticket fields contained null bytes (`\x00`) or very long text that could slow or break processing.
  - **Why it happened:** Real-world exports often contain hidden characters and large message bodies.
  - **How it was solved:**
    - Sanitized text by removing null bytes and trimming length before calling the LLM.
    - Built a "safe combined text" field using `subject + body` but with controlled limits.
  - **Outcome:** More stable ticket processing and fewer random failures.
- **Challenge 3 — Dataset variability (columns differ across CSVs)**
  - **What happened:** Different ticket datasets may not have the same column names (some have `subject/body`, others have `text/description`).
  - **Why it happened:** CSV sources and export formats vary across tools.
  - **How it was solved:**
    - Implemented **fallback logic** to automatically pick the best text-like column if the `subject/body` is missing.
    - Added **Debug: Ticket columns** view to quickly confirm dataset structure.
  - **Outcome:** The app can accept a wider variety of ticket CSVs without code edits.
- **Challenge 4 — Joining data across multiple SaaS tables**
  - **What happened:** Metrics require linking subscriptions, accounts, usage records, support tickets, and churn events.
  - **Why it happened:** Each file represents a different domain and uses different keys/time fields.
  - **How it was solved:**
    - Enforced consistent types: converted IDs to strings, dates to datetime.
    - Validated required columns early and stopped execution with a clear error if missing.
    - Used a consistent customer definition (`customer_id = subscription_id`) and joined account-level metadata on `account_id`.
  - **Outcome:** Reliable joins and consistent customer-level scoring.
- **Challenge 5 — Users needing flexibility without changing code**
  - **What happened:** The ticket dataset dropdown only showed files inside `data/raw/tickets`, so trying a new dataset required copying files into that folder.
  - **Why it happened:** The original app design expected a fixed folder-based workflow.
  - **How it was solved:**
    - Added **Upload CSV mode** for tickets so a dataset can be uploaded directly in the UI.
    - Saved uploaded datasets into the tickets folder so they become selectable and reusable.

    - **Outcome:** Much more user-friendly testing and demo workflow.
- **Challenge 6 — Streamlit rerun behavior and state resets**
    - **What happened:** Streamlit reruns the script on every interaction, which can reset computed results.
    - **Why it happened:** Streamlit's execution model is reactive.
    - **How it was solved:**
        - Used `st.session_state` to store enriched LLM results and last run timestamps.
        - Added a **Clear** button to intentionally reset state.
    - **Outcome:** The app keeps results stable across clicks and filters.

Future Improvements

- **Improve ticket-to-customer linkage (make LLM insights truly actionable)**
  - Add real mapping using shared keys (account_id, user email domain, CRM IDs).
  - Integrate with Zendesk/Salesforce exports to make joins accurate.
- **Move from rule-based scoring to a hybrid model**
  - Keep explainable rules for transparency, but add an ML model trained on churn labels.
  - Compare rule score vs model score; use both for better confidence.
- **Add time-series tracking**
  - Add “risk_score over time” trend per customer.
  - Track “usage drop % trend” and “ticket volume trend” week-over-week.
- **Speed + reliability improvements for LLM**
  - Add caching so repeated tickets are not reprocessed.
  - Add background queue / async processing for smoother UI.
  - Support GPU inference if available for faster throughput.
- **Production-style outputs**
  - Export customer playbooks (PDF/CSV), not just tables.
  - Add one-click “Create CRM note” / “Create task list” export formats.
- **Better governance**
  - Add a configuration file for thresholds and scoring weights.
  - Add an audit log of which rules fired for each customer.