

International Journal Research Publication Analysis

Page: 01-17

PRODUCTLENS AI: FORECASTING USER DISENGAGEMENT AND FEATURE SURVIVAL TO GROUND PRODUCT ROADMAP DECISIONS

***Aditya Malleli, Dr. M. Ramchandar (Guide)**

*Department of MCA (Master of Computer Applications) Chaitanya Bharathi Institute of
Technology (CBIT), Gandipet, Hyderabad, Telangana, India.*

Article Received: 05 September 2026

Article Revised: 25 September 2026

Published on: 15 October 2026

***Corresponding Author: Aditya Malleli**

Department of MCA (Master of Computer Applications) Chaitanya Bharathi
Institute of Technology (CBIT), Gandipet, Hyderabad, Telangana, India.

DOI: <https://doi-doi.org/101555/ijrpa.2474>

ABSTRACT

Product teams tend to learn that a feature has failed, or that a cohort is drifting away, only after a weekly dashboard has already turned red. This paper describes ProductLens AI, a modular pipeline that tries to bring that moment forward. Working on a simulated clickstream of 4,000 users and ten feature launches, the system combines leading engagement indicators, a nested conversion funnel, a gradient-boosted classifier that forecasts 14-day inactivity, Kaplan–Meier survival curves for feature usage, and a Beta–Bernoulli bandit that ranks candidate features by expected adoption per engineering hour. A language model then turns the computed outputs into a written roadmap recommendation and is not allowed to introduce figures of its own. Under a time-based holdout the classifier reaches an AUC of 0.977 overall, but only 0.792 among users who were active in the preceding week, which we argue is the fairer measure of early-warning value; a plain recency rule, by comparison, reaches 0.964 overall and only 0.632 on that subset. Survival curves separated the ten launches into sustained, novelty-driven and failed groups (day-14 survival of 0.72–0.79, 0.45–0.58 and 0.14–0.26), even though early adoption rates alone did not. The paper also shows that the practical size of the alert list depends heavily on the score threshold chosen, so that choice belongs with the product team and not with the model. We describe the architecture, the methods, the limits that come from synthetic data and heuristic posteriors, and a validation plan that should be completed before the pipeline is trusted on production telemetry.

INDEX TERMS: Product Analytics, Churn Prediction, Survival Analysis, Kaplan–Meier

Estimator, Multi-Armed Bandit, Feature Prioritisation, Large Language Models, Streamlit.

1. INTRODUCTION

Anyone who has sat through a weekly product review knows the pattern. A retention chart dips, somebody asks when it started, and the honest answer is a few weeks ago. By the time a feature is called a disappointment, its keenest early users have usually moved on, and next quarter's plan has already been drawn around it. Standard product analytics is very good at describing what happened. It is much less help with three questions a product manager needs answered before committing engineers: which of today's active users are drifting away, whether a freshly launched feature will stick, and what to build next given what earlier launches taught the team.

These questions usually belong to different people with different tools. Churn models are built by a data science group and rarely see feature-level data. Feature reviews lean on adoption counts or a fixed-day retention figure, and both quietly mishandle users who joined too recently to be observed for the whole window. Prioritisation is often a scoring workshop in which the estimates that matter most, namely how an idea will be received, are the ones with the least evidence behind them. Each practice is defensible alone. Together they leave gaps that nobody owns.

ProductLens AI is our attempt to close those gaps in one traceable pipeline. Every stage is framed as a forecasting problem, and each stage hands measured quantities to the next so that the final recommendation can be followed back to numbers. We built and tested it on a simulated clickstream in which the fate of every feature is known in advance. That lets us ask whether each detector recovers what was planted, instead of only whether its output looks plausible. The paper makes five contributions. First, an end-to-end architecture that links leading indicators, funnel and cohort analysis, disengagement forecasting, feature survival, backlog ranking and written synthesis, with a dashboard on top. Second, a two-tier evaluation of the disengagement model: besides a time-based holdout, we score only users who were active in the previous seven days, which separates early warning from the detection of people who have already left. Third, an application of censored survival analysis to feature usage, with evidence that early adoption and later survival can disagree. Fourth, a category-level Beta-Bernoulli bandit for ranking candidate features when history is short, including an explicit rule for categories never tried before. Fifth, a grounding scheme in which a language model writes the roadmap narrative from pipeline outputs alone. Section 2 reviews related work, Section 3 describes the system and data, Section 4 sets out the methods, Section 5

reports results, and Sections 6 to 8 give the discussion, limitations and conclusion.

2. RELATED WORK

Churn prediction has a long history in subscription businesses. Neslin *et al.* [1] examined the results of a large churn-modelling tournament to see which modelling choices drive predictive accuracy, and Verbeke *et al.* [2] argued for comprehensible models, using rule-induction methods on telecom data. Tree ensembles trained by gradient boosting [3] are now a common default for tabular behavioural features, and we use one. Evaluation on time-ordered data needs care in how training and test sets are drawn [4], so we split by date. Because inactivity is a minority outcome among the users who matter most, we also report average precision next to

ROC-AUC, following the argument that ROC curves can look reassuring on skewed data [5].

Survival analysis exists for observations that are cut off before the event of interest happens. The Kaplan–Meier estimator [6] is the standard non-parametric tool, and the Cox model [7] adds covariates. In marketing these methods are usually pointed at customers, asking how long a subscriber lasts, and cohort-based retention projection [8] is a close relative. We apply the estimator instead to the lifetime of one user’s use of one feature, so every launch gets its own curve. Curves were fitted with the lifelines package [9].

The multi-armed bandit problem goes back to Thompson’s 1933 paper on choosing between two treatments [10]. Upper-confidence-bound rules [11] and Thompson sampling [12] are the two families most often used, and contextual variants have been deployed for content recommendation at scale [13]. In product development the nearest counterpart is online controlled experimentation [14]. Our use is more modest. We allocate no live traffic; we borrow the posterior-plus-optimism logic to order a backlog whose entries have very little history.

Large language models [15] write fluently, which is exactly the risk when a topic is numerical, because fluent text can contain figures nobody computed [16]. Placing retrieved evidence in the prompt is a common mitigation [17]. ProductLens applies the idea in its plainest form: all computed results go in, and the model is told not to add any others.

3. SYSTEM ARCHITECTURE AND DATA

3.1 PIPELINE ARCHITECTURE

ProductLens is a set of seven Python modules run in sequence, each reading files written by

the earlier ones (Fig. 1). Handing results on as plain CSV, JSON and serialised-model files was deliberate. Any intermediate number can be opened and checked by hand, which matters when a written recommendation later has to be defended. A Streamlit dashboard reads the same files and presents them in five tabs: funnel and activation, cohort retention, at-risk users, feature survival with prioritisation, and the generated roadmap.

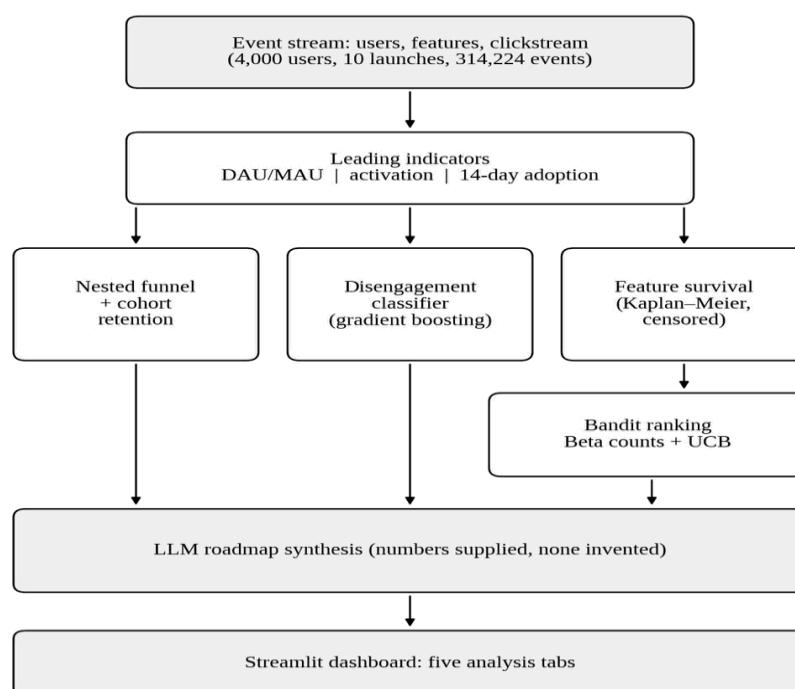


Fig. 1. ProductLens AI system architecture. Arrows show which stage consumes which stage’s output files.

3.2SIMULATED TELEMETRY

Event logs from a real product with known launch outcomes are hard to obtain and carry privacy duties, so we generated a Mixpanel-style event stream instead. With a fixed random seed, the simulator creates 4,000 users who sign up across 30 weekly cohorts between 1 January and 31 July 2025, with weekly signup weights rising from 0.6 to 1.4 to imitate growth. Each user gets one of five archetypes (Table 1) that fixes a weekly session range, a weekly chance of leaving, and a willingness to try new features. Churn is never logged as an event. A user simply falls silent, as in real telemetry, except that a “resurrector” who reaches a churn draw has an even chance of pausing for three to seven weeks and coming back.

Table 1: Simulated User Archetypes.

Archetype	Users	Sessions/wk	Churn hazard/wk	Try-feature prob.
power	475	5–9	0.010	0.75
regular	1,320	2–4	0.035	0.45
casual	1,243	0–2	0.080	0.25
one and done	556	0–1	0.350	0.10
resurrector	406	0–3	0.150	0.30

Target population shares were 12, 33, 30, 15 and 10%; user counts are the realised draw.

Ten features launch between week 3 and week 26. Each carries a hidden outcome: five “hits”, three “novelty-fade” features and two “flops”. Once a user has tried a feature, the chance of using it again in a session depends on the outcome class and on the weeks since that user’s first try. It declines slowly for hits and stops at 0.35, shrinks by a quarter each week for novelty features (floor 0.03), and shrinks by almost half each week for flops (floor 0.01). The result is a log of 314,224 events of eight types: sign-up, app open, five core actions, and feature use. The estimation steps never use the hidden labels; they are carried along only so that results can be compared with the truth afterwards.

4. METHODOLOGY

4.1 LEADING INDICATORS

Three indicators are computed first, all meant to move before retention curves do. Stickiness is the ratio of daily to trailing 30-day active users (Eq. 1). Activation for a signup cohort is the share of its members who perform at least one core action, feature use included, within seven days of signing up. The 14-day adoption rate of a feature is the share of users active in the two weeks after launch who tried it in that window. Limiting the denominator to people who were around to see the feature avoids diluting the rate with users who had already left.

$$DAU/MAU_t = |U_t| / |\cup_{d=t-29\dots t} U_d| \quad (1)$$

where U_d is the set of users with at least one event on day d .

4.2 FUNNEL AND COHORT RETENTION

The funnel has five steps: signing up, returning more than half a day later, activating, trying a feature within 14 days, and being active in both the second and third weeks. Each step is intersected with the one before, so the sets are strictly nested and every conversion figure is a true fraction of the previous step. A supplementary count sits outside the funnel: how many retained users never tried a feature in their first two weeks. It shows how far the core loop can keep people on its own. Cohort retention records, for each weekly cohort, the share of

members active in each later week. Cells for weeks that had not yet finished when the data ended stay empty instead of being filled with zero, so recent cohorts are not mistaken for weak ones.

4.3 DISENGAGEMENT FORECASTING

Each user contributes one snapshot per week, starting 14 days after signup and ending 14 days before the last logged event so that the outcome window is fully observed. Features at snapshot time t use earlier events only: recency (days since the last event); app-open counts over the trailing 7, 14 and 30 days; session trend (last week's opens minus the week before); core actions in the trailing 14 days; the number of distinct features used in 30 days; tenure; and the archetype, one-hot encoded. The label looks forward. It equals 1 if the user logs no event at all in the next 14 days (Eq. 2), where $N_u(a, b]$ counts user u 's events in that interval. The process gave 40,067 snapshots, 49.0% of them positive.

$$y_{u,t} = 1[N_u(t, t + 14 \text{ d}] = 0 \quad (2)$$

We trained a gradient-boosted tree classifier [3] with scikit-learn [18] (150 trees, depth 3, learning rate 0.08). Snapshots up to the 75th percentile of snapshot dates (25 June 2025) form the training set of 30,211 rows; later ones, 9,856 rows from 26 June to 17 July, form the test set. Besides ROC-AUC on the whole test set we run a second evaluation on snapshots whose recency is at most seven days (4,432 rows). A user who has been silent for a fortnight is easy to flag; the interesting question is whether the model can tell which currently active users will go quiet. For live scoring, every user with at least 14 days of tenure is scored at the end of the data, and those active in the last seven days are marked “actionable”, because only they can still be reached by an in-product message.

4.4 FEATURE SURVIVAL

For each feature and each user who ever used it, we measure how long that use lasts. Let f_u be the first use, l_u the last use, and $h_u = \min(T_{\text{end}}, f_u + 45 \text{ d})$ the end of the observation horizon. If $h_u - l_u \geq 14$ days, the usage is said to have died, with duration $l_u - f_u$. Otherwise the observation is censored at $h_u - f_u$: the user was still active, or we ran out of time to confirm a lapse. The Kaplan–Meier estimator [6] then gives the probability that usage has survived to day t (Eq. 3), where the product runs over all death times $t_i \leq t$, d_i is the number of deaths at t_i , and n_i is the number still at risk.

$$\hat{S}(t) = \prod_k (1 - d_k / n_k) \quad (3)$$

We report $\hat{S}$ at days 7, 14 and 30, the median lifetime, and pooled curves for each outcome

class. A user who tries a feature once and never returns is recorded as a death at day 0, so the curves describe the lifetime of use since first trial, one-off trials included.

4.5 BACKLOG PRIORITISATION

A regression from feature attributes to success would be hopeless with ten launches, so evidence is pooled by category (core workflow, personalisation, gamification, and so on). Each past launch f gets a sustained-adoption score $s_f = a_f \cdot \hat{S}_f(14)$, where a_f is its 14-day adoption rate. A feature therefore scores well only if people tried it and kept using it. Treating s_f as the success probability of each of the feature's n_f adopters, a category c starts from a uniform Beta(1, 1) prior and accumulates pseudo-counts (Eq. 4).

$$\alpha_c = 1 + \sum_{f \in c} s_f n_f \quad \beta_c = 1 + \sum_{f \in c} (1 - s_f) n_f \quad (4)$$

A candidate in category c is scored by an upper confidence bound (Eq. 5) and ranked by that bound per 100 estimated engineering hours h (Eq. 6), with $k = 1$.

$$UCB_c = \mu_c + k \sigma_c, \quad \mu_c = \alpha_c / (\alpha_c + \beta_c) \quad (5)$$

$$V = 100 \cdot UCB_c / h \quad (6)$$

Here σ_c is the standard deviation of the Beta(α_c , β_c) posterior. A candidate whose category has never shipped receives the prior itself, with mean 0.5 and standard deviation 0.289. That produces a large optimism bonus instead of a penalty for having no track record. The engineering-hour figures in the current implementation are illustrative stand-ins for real estimates.

4.6 ROADMAP SYNTHESIS

The last stage packs the funnel rows, per-cohort activation, the latest DAU/MAU value, the survival summary, the ranked candidates, the size of the actionable at-risk pool and the five strongest model signals into one JSON object. A system prompt casts the model as a senior product data strategist writing for a VP of Product, forbids any number that is not in the payload, and asks for five parts: an executive summary, what the data shows, survival findings, a ranked roadmap with a reason for each item (including how to treat the cold-start candidate), and one A/B test to validate the top recommendation. The request goes to a Claude Sonnet model

[19] with a 1,500-token limit. If no API key is configured, the script returns the fully assembled prompt instead, so the path from data to prompt can be inspected without the final

call. The intent is that the model contributes argument and not analysis.

5. RESULTS

Table 2 summarises every headline number reported below. Each value is derived in the subsection that follows, and the file it comes from is named; rows marked as supplementary were run in addition to the shipped pipeline, on the same data and split.

Table 2: Summary of Headline Results.

Metric	Value	Source
Simulated population	4,000 users; 314,224 events	users.csv, events.csv
Funnel: signup to retained (weeks 2 and 3)	39.6% (1,582 users)	funnel.csv
Largest funnel step: activated to tried a feature	64.2%	funnel.csv
Disengagement AUC, all test snapshots	0.977	disengagement_snapshots.csv
Disengagement AUC, still-active subset (AP)	0.792 (0.242)	disengagement_snapshots.csv
Recency-only AUC, still-active subset	0.632	supplementary run
Day-14 survival: hits / novelty / flops	0.72–0.79 / 0.45–0.58 / 0.14–0.26	feature_survival_summary.csv
Top-ranked candidate (value per 100 h)	custom_themes (0.473)	feature_prioritization_ranking.csv
Active users with risk score of 0.20 or more	155 of 1,556	at_risk_users.csv

5.1 FUNNEL AND RETENTION

Of 4,000 signups, 3,665 (91.6%) returned, 3,156 of those activated (86.1%), 2,027 tried a feature within two weeks (64.2%), and 1,582 (78.0% of the previous step, 39.6% of signups) were active in both weeks 2 and 3 (Fig. 2). The step from activation to a first feature trial is the biggest leak: 1,129 activated users never touched a feature in their first fortnight. Features are not a precondition for staying, though. Without the nesting requirement, 2,171 users were active in both weeks, and 549 of them (25.3%) had not tried any feature.

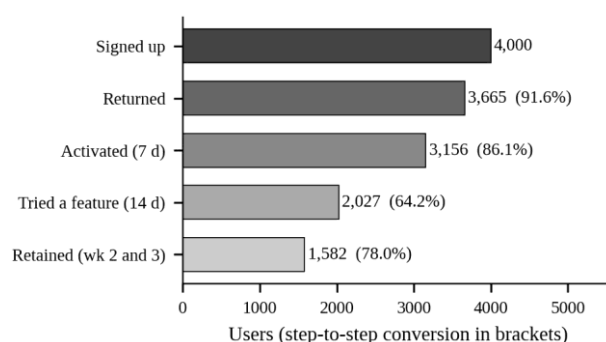


Fig. 2. Nested engagement funnel. Percentages give conversion from the preceding step.

Averaged across cohorts, weekly activity falls from 73.1% in week 1 to 58.9% in week 4 and 40.3% in week 12 (Fig. 3). Cohort activation rates lie between 70.8% and 87.9%, and DAU/MAU has been steady, with a median of 0.27 and a last value of 0.268 on 31 July. The heatmap shows a smooth and broadly similar decay in every cohort, which is expected here because the simulator draws the same archetype mix for each.

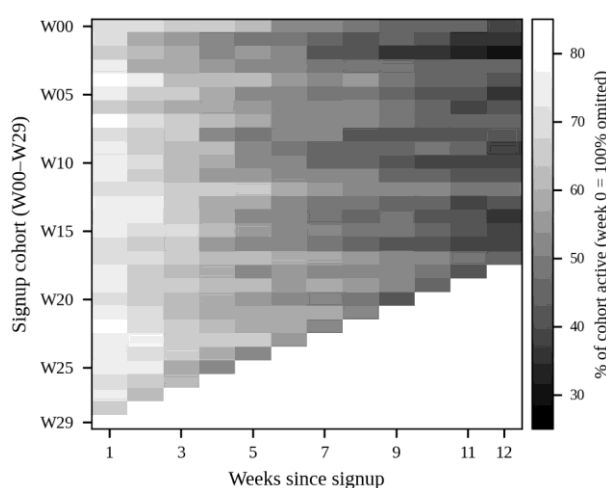


Fig. 3. Weekly cohort retention. Empty cells are weeks not yet observed.

5.2 DISENGAGEMENT FORECASTING

Table 3 compares the deployed model with simpler alternatives, all trained and tested on the same time split. On the whole test set every model looks excellent, and the raw recency column alone reaches an AUC of 0.964 [20]. That is a property of the label: a user who has been silent for two weeks will most likely stay silent for the next two. At a 0.5 threshold the boosted model's precision and recall on the full test set are 0.946 and 0.916 (Appendix A). The picture changes on the still-active subset (Fig. 4), where inactivity occurs in only 7.5% of snapshots. Recency alone falls to 0.632, logistic regression reaches 0.735, and the boosted model reaches 0.792 with an average precision of 0.242 against a base rate of 0.075. Within

that subset, the 10% of snapshots rated riskiest contain inactivity 22.8% of the time, a 3.0-fold lift.

Table 3: Disengagement Models on the Time-Based Test Set.

Model	AUC, all	AUC, active	AP, active
Recency alone (no model)	0.964	0.632	0.136
Logistic regression	0.972	0.735	0.220
Gradient boosting (deployed)	0.977	0.792	0.242
Gradient boosting, no recency	0.976	0.795	0.246
Logistic regression, no recency	0.968	0.723	0.192

“Active” = snapshots with recency ≤ 7 days ($n = 4,432$; 7.5% positive). AP = average precision. Rows 1, 2, 4 and 5 were run in addition to the deployed pipeline, on the same split.

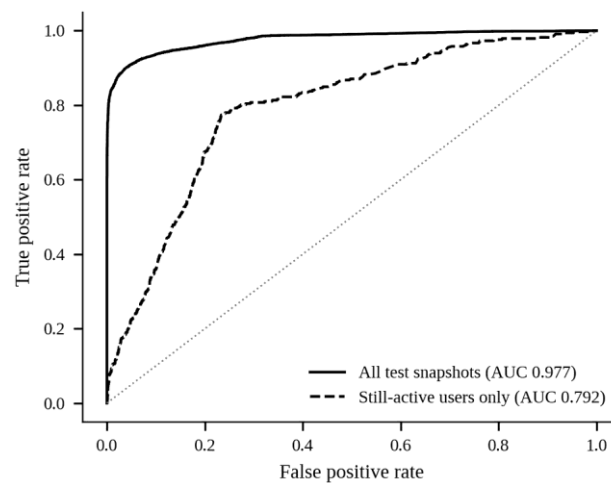


Fig. 4. ROC curves of the deployed model on all test snapshots and on users who were active in the previous seven days.

In the fitted model recency carries 92% of the total feature importance (Appendix C). To see whether the model really depends on it, we retrained without that column. The AUC barely moved (0.976 on the full set, 0.795 on the active subset), so session and core-action counts hold much the same information. The importance figure describes how the trees split, and it does not show that recency is indispensable.

The threshold matters a great deal for the still-active users. At 0.5 the model raises only 46 alerts among 4,432 snapshots; 25 of them are correct, which is a precision of 54% but a recall of only 7.5% (Appendix A). Lowering the threshold to

0.20 gives 490 alerts with 23.1% precision and 34.0% recall, and at 0.10 there are 1,216 alerts that capture 77.4% of the users who later lapse, at 21.1% precision. Scoring the 3,666 users with at least 14 days of tenure at the end of the data leaves 1,556 (42.4%) still active. Their

risk scores are low on the whole (median 0.041, mean 0.081): 444 exceed 0.10, 155 exceed 0.20 and 15 exceed 0.30. The list of users worth contacting is therefore a short tail and not the whole active pool, and the cut-off has to be chosen against the cost of an intervention.

5.3 FEATURE SURVIVAL

Table 4 lists the estimates. The five hits keep 72–79% of their adopters at day 14 and 57–68% at day 30, and none reaches a median lifetime within the horizon. The novelty features hold 45–58% at day 14. The two that were followed long enough, `gamified_streaks` and `confetti_animations`, are down to 28% and 24% by day 30, with median lifetimes of 15 and 12 days. The two flops keep only 26% (`voice_input`) and 14% (`social_feed`) at day 14, and their median is zero, meaning that more than half of their adopters never used them a second time. The pooled curves in Fig. 5 tell the same story.

Table 4: Kaplan–Meier Feature Survival.

Feature	Class	n	$\hat{S}(7)$	$\hat{S}(14)$	$\hat{S}(30)$
ai recommendations	H	1,373	0.825	0.790	0.680
smart search v2	H	1,572	0.830	0.778	0.669
dark mode	H	1,551	0.828	0.774	0.661
bulk export	H	1,143	0.791	0.733	0.643
team workspaces	H	913	0.778	0.720	0.575
quick actions menu	N	483	0.645	0.583	0.583†
gamified streaks	N	1,422	0.625	0.508	0.282
confetti animations	N	995	0.580	0.450	0.242
voice input	F	388	0.360	0.259	0.125
social feed	F	820	0.235	0.139	0.066

H = hit, N = novelty fade, F = flop; n = adopters. †No adopter had been followed for 30 days, so the value repeats the last observable one.

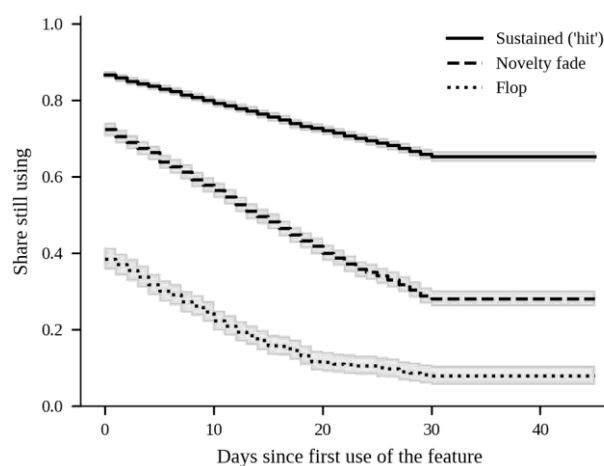


Fig. 5. Pooled Kaplan–Meier curves by outcome class with 95% confidence bands.

Fig. 6 sets early adoption against later survival, and adoption alone would have misled in several cases. `ai_recommendations`, a hit, was tried by 24.5% of eligible users, fewer than `gamified_streaks` (29.5%, a novelty fade), yet its day-14 survival was 79% against 51%. The 13.2% adoption of `social_feed` was close to `team_workspaces` (15.4%) and `bulk_export` (18.1%), but only 14% of its adopters kept using it, against 72% and 73%. Two properties of the curves deserve a warning. They flatten after about day 30 because, with a 45-day horizon and a 14-day gap rule, a lapse can only be confirmed for users who stopped before about day 31. And the drop at day 0 comes from one-off trials.

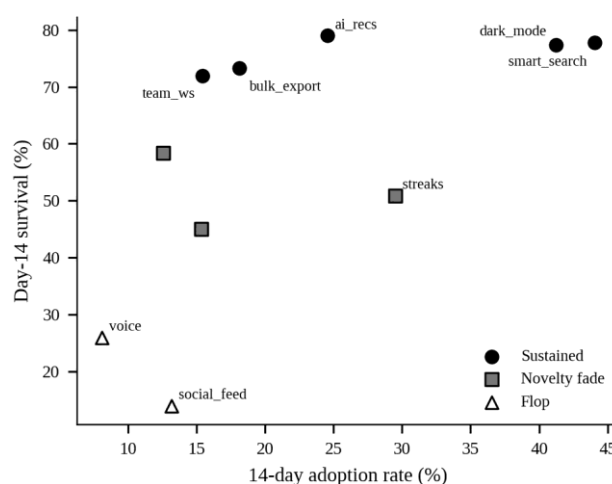


Fig. 6. Fourteen-day adoption rate against day-14 survival for the ten launches.

5.4PRIORITISATION

Category posterior means are 0.320 for personalisation, 0.273 for core workflow, 0.117 for gamification, 0.116 for productivity, 0.112 for collaboration and 0.019 for social (Appendix B). Table 5 gives the ranking they produce. `custom_themes` comes first because it belongs to the strongest category and costs only 70 hours. `offline_mode`, from a category the product has never shipped, comes second on the strength of its exploration bonus. Its upper bound of 0.789 is the highest on the list, but at 400 hours cost pulls it down, and with the bonus switched off ($k = 0$) it would sit fourth, behind `advanced_filters` and `keyboard_shortcuts`. The two core-workflow candidates share a posterior, so they differ only in cost, and `ai_chat_assistant` costs 3.3 times as much as `advanced_filters` for the same expected benefit. The social and collaboration candidates end up at the bottom.

Table 5: Candidate Features Ranked by Value per 100 Hours.

Candidate	Category	Hours	UCB V (Eq. 6)
custom_themes	personalization	70	0.3310.473
offline_mode*	infrastructure	400	0.7890.197
advanced_filters	core_workflow	150	0.2820.188
keyboard_shortcuts	productivity	90	0.1240.137
leaderboards	gamification	110	0.1240.112
ai_chat_assistant	core_workflow	500	0.2820.056
comments_and_mentions	collaboration	280	0.1220.044
public_profile_sharing	social	260	0.0240.009

*Cold-start candidate scored with the Beta(1, 1) prior.

5.5 ROADMAP SYNTHESIS AND DASHBOARD

We did not run a live model call for this paper. The repository stores the assembled prompt together with an illustrative roadmap that was written by hand from the pipeline's saved numbers to show the intended format. It is not model output and we do not evaluate it. What can be said is that its ordering agrees with Table 5. It recommends custom_themes first, treats offline_mode as a scoped beta and not a full build, and proposes a two-week A/B test of a lightweight version of the leading candidate before engineering time is committed. The dashboard exposes every stage described above, including a table of the fifty highest-risk active users and a control for regenerating the roadmap when a key is supplied.

6. DISCUSSION

Three findings seem more broadly useful than the particular numbers. First, an evaluation over all snapshots rewards a churn model for noticing people who have already gone. A one-line recency rule scored 0.964 there, and the boosted model added only 0.013. Only the second tier, restricted to users who were still active, showed what the model contributes: a rise from 0.632 to 0.792. Any team that reports a single churn AUC should ask which of these two questions it answers.

Second, the early adoption rate is a weak guide to whether a feature will last. Across the ten launches the two are positively related (Spearman correlation 0.73), yet the relationship fails for exactly the cases a team cares about: a flop such as social_feed looked much like the hit team_workspaces on adoption alone. Survival estimates that respect censoring separated the classes cleanly and can be computed within weeks of launch.

Third, the usefulness of a risk score is set by the alert threshold as much as by the model. In the still-active subset a cut-off of 0.5 produced 46 alerts and found under a tenth of the users who would lapse, while a cut-off of 0.10 found three quarters of them at the price of about four

false alarms for every correct one. That trade-off depends on what an intervention costs, so it is a decision for the product team and should be reported next to the AUC.

7. LIMITATIONS

Synthetic data. Every result comes from data whose generating rules we wrote. That survival curves separate hits, fades and flops shows the estimator can recover a planted structure. It says nothing about how sharply real features would separate. The 0.79 subset AUC likewise rests on the simulator's assumption that activity and departure follow simple archetype-specific processes, while real users bring seasonality, notifications, outages and pricing changes. Cohorts differ only in size in our simulation, so the heatmap says little about cohort quality.

Forecasting target. The label counts any event as activity, whereas a product might prefer to count only valuable actions, and the 14-day window is a convention. The model is uncalibrated, so scores should be read as a ranking. The "actionable" pool is the set of still-active users and not an alarm list.

Bandit posteriors. Treating a product of two rates as a success probability, and each adopter as an independent trial, yields posterior standard deviations near 0.01 for categories with data. That is much tighter than ten launches justify, because launches within a category are not independent and most categories hold one or two of them. The ranking is best read as a transparent scoring rule with an exploration term and not as calibrated uncertainty. A hierarchical model that shrinks categories toward a shared mean is the natural next step. Value is measured in adoption, not revenue or strategic fit, and the hour estimates are placeholders.

Survival. The 14-day gap and 45-day horizon are choices, and recent launches have short follow-up, so their later estimates are extrapolated by construction. The estimates describe association. Whether a feature *causes* better retention needs a controlled experiment [14].

Language-model layer. The layer is not evaluated quantitatively here. A sensible safeguard would be to extract every numeric token from the generated text and check it against the input JSON before the document reaches a reader, alongside human review. Scoring individual users also raises privacy questions; we would restrict such scores to in-product assistance.

8. CONCLUSION AND FUTURE WORK

ProductLens AI shows one way to connect churn forecasting, feature survival and backlog ranking into a single chain of evidence. On simulated data it recovered the planted structure: a

classifier whose value shows up only when evaluated on users who are still active, survival curves that expose feature failures which adoption rates hide, and a bandit ranking that treats untried categories as uncertain instead of poor. Its weakest links are also visible, namely the synthetic data, the heuristic posteriors and the unevaluated language-model layer. Future work should run the pipeline on production telemetry, replace the category posteriors with a hierarchical model, add covariates through Cox regression, calibrate the risk scores and estimate uplift for interventions, check generated text against its input numbers automatically, and close the loop with live A/B tests of the top-ranked candidates.

APPENDIX A — CONFUSION MATRICES, DEPLOYED CLASSIFIER

On the time-based test set, with a decision threshold of 0.5 and “inactive within 14 days” as the positive class:

Table 6: Confusion Counts and Precision/Recall at Threshold 0.5.

Subset	TN	FP	FN	TP	Precision	Recall
All test snapshots (n = 9,856)	4,329	274	441	4,812	94.6%	91.6%
Still-active subset (n = 4,432)	4,079	21	307	25	54.3%	7.5%

On all snapshots the model is both precise and sensitive, largely because long-silent users are easy cases. Among still-active users it is precise when it speaks but speaks rarely at this threshold (Section 5.2).

APPENDIX B — CATEGORY POSTERIORS

Table 7: Beta Posteriors Accumulated from the Ten Launches.

Category	Launches	α	β	Mean	Std.
personalization	1	496.3	1,056.7	0.3196	0.0118
core_workflow	2	805.6	2,141.4	0.2734	0.0082
gamification	2	282.9	2,136.1	0.1169	0.0065
productivity	2	188.2	1,439.8	0.1156	0.0079
collaboration	1	102.3	812.7	0.1119	0.0104
input_method	1	9.1	380.9	0.0234	0.0076
social	1	16.0	806.0	0.0194	0.0048
infrastructure (no history)	0	1.0	1.0	0.5000	0.2887

Pseudo-counts follow Eq. 4. The last row is the uninformative prior used for the cold-start candidate.

APPENDIX C — FEATURE IMPORTANCE, DEPLOYED CLASSIFIER

Table 8: Impurity-Based Importance of Model Inputs.

Input	Importance
recency days	0.9213
archetype casual	0.0230
archetype one and done	0.0117
sessions 30d	0.0092
sessions 14d	0.0087
archetype regular	0.0086
archetype resurrector	0.0077
sessions 7d	0.0041
core actions 14d	0.0036
session trend	0.0011
tenure days	0.0009
feature breadth 30d	0.0001
archetype power	0.0000

Values are the gradient-boosting model's built-in importances and sum to one.

REFERENCES

1. S. A. Neslin, S. Gupta, W. Kamakura, J. Lu, and C. H. Mason, "Defection detection: Measuring and understanding the predictive accuracy of customer churn models," *J. Marketing Res.*, vol. 43, no. 2, pp. 204–211, May 2006.
2. W. Verbeke, D. Martens, C. Mues, and B. Baesens, "Building comprehensible customer churn prediction models with advanced rule induction techniques," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2354–2364, Mar. 2011.
3. J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.
4. C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Inf. Sci.*, vol. 191, pp. 192–213, May 2012.
5. T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, Art. no. e0118432, 2015.
6. E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *J. Amer. Statist. Assoc.*, vol. 53, no. 282, pp. 457–481, 1958.
7. D. R. Cox, "Regression models and life-tables," *J. Roy. Statist. Soc. B*, vol. 34, no. 2, pp. 187–220, 1972.
8. P. S. Fader and B. G. S. Hardie, "How to project customer retention," *J. Interactive Marketing*, vol. 21, no. 1, pp. 76–90, 2007.

9. C. Davidson-Pilon, “lifelines: Survival analysis in Python,” *J. Open Source Softw.*, vol. 4, no. 40, p. 1317, 2019.
10. W. R. Thompson, “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples,” *Biometrika*, vol. 25, nos. 3–4, pp. 285–294, 1933.
11. P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Mach. Learn.*, vol. 47, nos. 2–3, pp. 235–256, 2002.
12. D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, “A tutorial on Thompson sampling,” *Found. Trends Mach. Learn.*, vol. 11, no. 1, pp. 1–96, 2018.
13. L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proc. 19th Int. Conf. World Wide Web (WWW)*, Raleigh, NC, USA, 2010, pp. 661–670.
14. R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
15. T. B. Brown *et al.*, “Language models are few-shot learners,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 1877–1901.
16. Z. Ji *et al.*, “Survey of hallucination in natural language generation,” *ACM Comput. Surv.*, vol. 55, no. 12, Art. no. 248, 2023.
17. P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Proc. NeurIPS*, vol. 33, 2020, pp. 9459–9474.
18. F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
19. Anthropic, “Claude API documentation,” 2026. [Online]. Available: <https://docs.claude.com>
20. T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.